An overview of hardware security and trust : threats, countermeasures and design tools

Hu, Wei; Chang, Chip-Hong; Sengupta, Anirban; Bhunia, Swarup; Kastner, Ryan; Li, Hai

2020

Hu, W., Chang, C., Sengupta, A., Bhunia, S., Kastner, R. & Li, H. (2020). An overview of hardware security and trust : threats, countermeasures and design tools. IEEE Transactions On Computer-Aided Design of Integrated Circuits and Systems. https://dx.doi.org/10.1109/TCAD.2020.3047976

https://hdl.handle.net/10356/147019

https://doi.org/10.1109/TCAD.2020.3047976

An Overview of Hardware Security and Trust: Threats, Countermeasures and Design Tools

Wei Hu, *Member, IEEE,* Chip-Hong Chang, *Fellow, IEEE,* Anirban Sengupta, *Senior Member, IEEE,* Swarup Bhunia, *Senior Member, IEEE,* Ryan Kastner, *Senior Member, IEEE,* and Hai Li, *Fellow, IEEE*

(Invited Paper)

***Abstract*—Hardware security and trust have become a pressing issue during the last two decades due to the globalization of the semi-conductor supply chain and ubiquitous network connection of computing devices. Computing hardware is now an attractive attack surface for launching powerful cross-layer security attacks, allowing attackers to infer secret information, hijack control flow, compromise system root-of-trust, steal intellectual property (IP) and fool machine learners. On the other hand, security practitioners have been making tremendous efforts in developing protection techniques and design tools to detect hardware vulnerabilities and fortify hardware design against various known hardware attacks. This paper presents an overview of hardware security and trust from the perspectives of threats, countermeasures and design tools. By introducing the most recent advances in hardware security research and developments, we aim to motivate hardware designers and electronic design automation tool developers to consider the new challenges and opportunities of incorporating an additional dimension of security into robust hardware design, testing and verification.**

***Index Terms*—Hardware security, security threat, security countermeasures, design tools, survey.**

I. Introduction

MODERN computing hardware devices are usually crafted by vendors with different established levels of trust and at discrete locations. These hardware components, while residing in a mixed-trust computing environment, are often shared among execution contexts of different security levels in a back-to-back manner. In addition, the rich connectivity features of modern computing systems expose critical hardware resources to attackers and open up doors for remote attacks without requiring physical access to the victim. As a consequence, our computing hardware is ever closer to the front-line of a burning battle field and confronted with various security threats.

Hardware security threats can arise during various stages of the entire semiconductor life cycle, ranging from specification to fabrication and even recycling. They can result from unintentional design flaws [1]–[3], system side effects [4]–[7] and intended malicious design modifications [8]–[10]. They usually target security assets like cryptographic functions, secure architectures, intellectual property (IP) and machine learning (ML) models. While classic hardware security threats such as covert and side channels, hardware Trojans and reverse engineering (RE) are constantly evolving, recent powerful attacks exploit remote [6], [11], cross-layer [2], [3], [12], specification-compatible [8], [13] attack surfaces to compromise strong cryptographic primitives, isolation mechanisms, memory protection techniques and deep neural networks (DNNs). Understanding the different hardware security threats is an important first step to developing effective security countermeasures and design tools for circumventing them.

Security practitioners have been making tremendous efforts in developing effective hardware security countermeasures. An important first task is to create hardware security primitives that can serve as the building blocks for crafting an architectural level trusted computing environment enhanced with strong isolation mechanisms. Effective side channel protection and Trojan detection techniques are essential for verifying that the security primitives and trusted computing environment are free of design flaws, covert and side channels, and backdoors. Recent advances in ML and artificial intelligence (AI) have shown promise in developing more accurate detection solutions [14], [15]. IP protection techniques [16]–[19], on the other hand, protect security primitives, hardware designs and DNN models from RE, counterfeiting, model extraction and other adversary attacks.

Despite the numerous protection techniques for thwarting hardware security threats, security is still at large an afterthought in hardware design. Most security holes are exposed only after their exploitation by the threat actors. Over-reliance on software patches for hardware flaws also contributed to the trove of zero-day exploits for the attackers. In many ways, the database of common vulnerabilities and exposures (CVE) is just the tip of an iceberg. This is largely due to the lack of effective hardware security tools that allow automated specification, verification and evaluation of security constraints.

Manuscript received June 13, 2020; revised October 3, 2020; accepted December 15, 2020. This paper was recommended by Associate Editor Y. Makris.

Corresponding author: Chip-Hong Chang. W. Hu and C. H. Chang contribute equally to this article. This research is supported in part by the National Research Foundation, Singapore, under its National Cybersecurity R&D Programme/Cyber-Hardware Forensic & Assurance Evaluation R&D Programme (NCR Award CHFA-GC1-AW01) and the National Natural Science Foundation of China under grant 62074131.

W. Hu is with the School of Cybersecurity, Northwestern Polytechnical University, China. E-mail: weihu@nwpu.edu.cn.

C. H. Chang is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. E-mail: echchang@ntu.edu.sg.

A. Sengupta is with the Discipline of Computer Science and Engineering at Indian Institute of Technology (IIT) Indore, India. E-mail: asengupt@iiti.ac.in.

S. Bhunia is with the Department of Electrical and Computer Engineering, University of Florida, USA. E-mail: swarup@ece.ufl.edu.

R. Kastner is with the Department of Computer Science and Engineering, University of California San Diego, USA. E-mail: kastner@ucsd.edu.

H. Li is with the Department of Electrical and Computer Engineering, Duke University, USA. E-mail: hai.li@duke.edu.

Best design practices and tacit knowledge are necessary but inadequate. We need better design tools to enforce hardware security properties for trust assurance. Proactive hardware information flow analyses facilitate vulnerability shielding and on-site monitoring. For example, unintentional hardware flaws and potential security vulnerabilities can be detected early by the recent security-driven hardware design flow [20]–[22]. As the rally of attacks and countermeasures is a never-ending recursion, it is important to keep abreast of its latest development to continuously close the productivity gap of secure hardware design. If the tool chain does not constantly update to catch up with the latest design-for-trust and security verification methodologies, at the present rate of growth in hardware design complexity, security may terminate Moore's law before other physical limits.

At present, the hardware security space has grown to a point with many different specialized topics and each topic has been discussed in several focused survey papers. Examples of recent surveys on a few specialized topics are side and covert channels [4], [23], [24], reverse engineering [25], hardware Trojan [26], [27], physical unclonable function [28], logic locking [29] and security verification tools [30]. This paper provides a concise overview of hardware security from three perspectives, namely threats, countermeasures and design tools, with emphasis on niche, uncharted topics and updated recent developments for hardware security in a mixed-trust environment. We also identify potential future research directions with this overarching vision.

The reminder of this paper is organized as follows. A brief description about the common hardware security properties is introduced in Section II. In Section III, an overview of the classic as well as state-of-the-art hardware security attacks is provided. Section IV reviews the frequently used hardware security mechanisms for thwarting these attacks. Section V summarizes various secure hardware design tools from both the academia and industry. Some research challenges and opportunities in the discussed topics are highlighted in Section VI and the paper is concluded in Section VII.

II. Hardware Security Properties

Hardware security properties are formal specifications about invariant security-related behaviors of circuit designs. Security threats and attacks usually cause violations of desirable security properties while security countermeasures implement mechanisms for enforcing them. Security properties provide important constraints to security verification tools. In the following, we briefly cover different hardware security properties.

A. Dependability

Reliability, *Availability* and *Safety* are three important attributes to assess the trustworthiness of a computing hardware device to perform the expected function during its service lifespan. Reliability is the ability to produce the intended functions under normal operation and even under small fluctuations in the computing environment for a specified time period. Availability is the percentage of time a system is able to serve its intended function. Safety is the ability to avoid catastrophic consequences for the user or the environment. Catastrophic failures represent only a small subset of all failures. Hence, safety is a relative and subjective attribute that cannot be measured directly. Critical path timing failures, single-event-upsets and aging effects can be resulted from security exploits, such as fault injection [31] and recent ML attacks [32], to reduce the reliability, increasing the downtime or impose safety hazards upon a system.

B. Confidentiality

Confidentiality is a general security property stating that secret information should never be obtained or inferred by observing a public output or memory location. While the direct movements of sensitive information can be easily identified, the stealthy leakage through system side effects and back-doors can be more subtle. These include the covert and side channels [4], [33] in cryptographic cores, system bus and high-performance elements such as caches and branch predictors [2], [3] as well as hardware Trojans [34].

C. Integrity

Integrity is the dual property of confidentiality. It requires that a trusted data object should never be overwritten by an untrusted entity. Integrity attacks often target critical memory locations, e.g., the cryptographic key, program counter and privilege registers. These attacks are usually a first step for performing further malicious activities, e.g., hijacking the control flow [35] and fooling machine learners [36].

D. Isolation

Isolation is a two-way property requiring that two hardware components of different security levels should not directly communicate with each other. It is a common security property that needs to be enforced in System-on-Chip (SoC), modern processors and the cloud, where the interaction between the secure and normal worlds are strictly controlled. However, there are still ingenious security exploits that break strong isolation mechanisms such as ARM Trust-Zone [37] and Intel Software Guide Extension (SGX) [38].

E. Constant Time

The *constant time* security property enforces that the hardware design should take invariant amount of time to compute and produce the result under different input combinations. In other words, we cannot learn any information about the inputs by observing the computation time. Violation of the constant time property creates a timing channel that can leak sensitive information. Such violation can result from performance optimizations [2], [3], e.g., cache and branch predictor as well as fast path in arithmetic units.

F. Quantitative Security Properties

Quantitative security properties allow more accurate measurement of hardware design security, e.g., assessing the severity of a vulnerability or evaluating the effectiveness of a security protection mechanism. Typical examples of such quantitative properties include randomness of the output of a cryptographic function [20], leakage of side and covert channels [39] and strength of a security mitigation [40]. These security properties are usually measured using statistical and information theoretic security metrics [41]. The security of approximate computing and machine learning are more often measured quantitatively.

III. Hardware Security Threats

A. Architectural and System Threats

1) Secure Boot Attacks: A secure boot starts by loading code from an immutable boot ROM, correctly initializing critical peripherals, configuring security and system settings, authenticating and properly loading boot images and application code and properly sanitizing data upon reset. Many issues arise due to the system being configured incorrectly, e.g., system memory space not protected. Other issues relate to data not being properly erased (e.g., keyboard strokes stay in buffers). These and many other real-world secure boot attacks are documented by Bulygin *et al.* [42].

The secure boot process is fairly well-documented making it amenable to formal property specification [43], [44]. Such properties relate to isolation and access control between boot stages (e.g., the next stage can only access a limited subset of the previous stage information), determining if a boot stage completes fully before continuing to the next stage and protecting boot state information properly upon completion (e.g., it cannot be modified and can only be read from boot code). Additionally, there should be a sequence that causes the hardware to fully reset all data, code, configuration and any other state, and the system should only load from the boot ROM upon reset.

2) Firmware Attacks: Firmware is the low-level software that controls the interaction and behavior of a piece of hardware or IP core. Firmware plays a key role in determining the security of the SoC. Incorrectly setting configuration registers can lead to catastrophic consequences and open the door to leaking confidential information, unsafe behaviors and critical flaws that can be exploited by attackers. An analysis in 2014 showed that at least 140,000 devices had a firmware vulnerability [45]. This should not be too surprising as determining the correctness of the firmware is challenging as each hardware core has different configurations that interact with the overall system in a non-obvious manner.

Firmware is particularly important for SoC architectures. Modern SoC architectures are a patchwork of hundreds, sometimes thousands, of different IP cores that are cobbled together from in-house sources, outside vendors and open source repositories. Ensuring that these are functionally correct is a massive undertaking; determining that they lack security flaws is even more challenging. Subramanyan *et al.* [46] provide good motivation and the early work in this space.

Device drivers are typically small, but important pieces of low-level C or assembly code that play an important role in firmware security. They provide an application program interface (API) that is used to deliver data to/from a device, query the status of the device, or set the device mode. More often than not, device drivers require access to critical parts of the system and thus it is crucial that they execute efficiently, handle real-time constraints and be secure. A first step towards synthesizing correct, efficient and secure device drivers is to create properties around on-chip communication protocols like Advanced Extensible Interface (AXI) and Wishbone [47]. Properly handling access control to the hardware resource is also important for secure computing with devices [48].

3) Dynamic Random Access Memory (DRAM) Threats: The Coldboot [49] and Rowhammer [50] attacks demonstrate the importance of protecting sensitive data stored in DRAM. Coldboot exploits the physical phenomenon that DRAM data persists for a short amount of time even after powering off the memory. This time can be extended by cooling down the memory, which further reduces the leakage of current from the DRAM capacitors. Researchers used this idea to show how to remove a DRAM from one computer, place it into another and grab the data. Other malicious attacks are also possible. Rowhammer exploits another physical vulnerability of DRAM, this time using the fact that DRAM data can be altered by accessing nearby data. The attacker locates some of their data next to some critical data in DRAM. By changing the values of their data, the attacker induces circuit noise that causes the target sensitive data to change.

4) Cache Attacks: Cache attacks [4] exploit information leakage though cache state and are extremely effective at extracting protected information. The cache is a shared resource and any process that uses it can leave traces about their computation, in particular, the memory addresses they accessed.

Cache timing attacks can be categorized as time-driven and access-driven [23]. A time-driven attack measures the execution time of the victim process. The attacker manipulates the contents of a shared cache and observes the timing of another process (e.g., a cryptographic operation). The timing is effected by cache hits and misses, which provides information about the key [51]. An access-driven attack extracts information by measuring the time that it takes the attacker to perform a cache access [52]. If a particular cache line is accessed by the victim process, the attacker would observe a cache hit and vice-versa. For instance, an attacker can identify data access patterns by the victim (e.g., which S-Box entries are being accessed during AES execution) and use this information to extract the confidential information (e.g., the secret key). Cache side channel is a powerful attack that is often used in combination with other attacks, e.g., Meltdown [3] and Spectre [2] as we will discuss.

5) Speculative Execution Attacks: Meltdown [3] and Spectre [2] are the first of a series of attacks that leverage speculative execution, out of order execution, caching and other architectural performance enhancements to break isolation and other security policies.

Meltdown enables unauthorized processes to read data from any address that is mapped to the current process's memory space. Meltdown exploits a race condition where the unauthorized process attempts to access privileged data. A privilege check eventually squashes the execution of that code, but not before the data is temporarily loaded into cache. The attack then uses a cache side-channel attack (SCA) to determine contents of the data.

Spectre is a vulnerability that tricks a victim process to leak its data. Many processors perform speculative execution by branch prediction. Spectre uses the fact that this speculative code leaves traces of its execution in the cache whose information can be extracted using a cache SCA (similar to Meltdown). Spectre trains a branch predictor to make a wrong decision and then wraps code that should not be executed in a condition. The code is speculatively executed since the branch predictor is wrong. It eventually gets squashed but it leaves important information in the cache state, which is extracted via a cache SCA.

6) Code Reuse Attacks: Code reuse attacks carefully use existing snippets of software to perform computation of the attackers choosing. Return oriented programming (ROP) [53] is an example of code reuse attack where existing code fragments (or gadgets) are carefully sequenced to perform a malicious act. The attacker's goal is to divert the control flow by gaining control of the call stack and invocating the first gadget, which in term calls subsequent gadgets. This allows the attacker to perform actions of their choosing.

B. Covert and Side Channels

Covert and side channels have emerged as two types of potent information leakage channels. Micro-architectural features targeted towards performance improvement, e.g., shared cache, speculative control, and hyper-threading, create new covert and side-channel security issues. Covert channels use non-traditional communication mechanisms to leak critical information – often between an insider process (e.g., a Trojan Horse program) and an outsider spy process. These two processes do not communicate directly through traditional mechanism, e.g., shared cache. Instead, a Trojan process may communicate with a spy process by modulating timing of specific events on a shared resource or writing/checking if a file is locked. On the other hand, SCAs utilize physical side-channel parameters (supply current, event timing, electromagnetic emission, etc.) to leak on-chip secrets.

While some covert channels require sharing of hardware resources among exchanging parties (e.g., shared cache), others may exist among hardware components that are physically isolated or not even in proximity. Hardware-oriented covert channels are typically initiated by introducing manipulation or exploitation of certain functional (response to a fault) or parametric (e.g., timing, power and electromagnetic radiation, etc.) behavior of the hardware that is observed to decode the secret information being transmitted. Side channels are unintentional information leakage where an attacker tries to extract information from a target computing system utilizing its inherent implementation vulnerabilities. Similar to covert channels, side channel also requires the observation of certain functional or parametric behaviors at runtime.

1) Timing Channel: A timing channel is established through the observation of the execution time of a certain process. Timing-based covert and side channels may exist due to the sharing of hardware resources across different software processes. Moreover, an IC may contain fast and slow execution paths that reveal information regarding the underlying operation being executed (e.g., arithmetic vs. Boolean operation [54]. While chip designers introduce novel features to improve execution time, more timing channels are being discovered. These channels may facilitate information transfer at a rate of up to few megabits per second [55]. These timing channels are often practical only under certain assumptions regarding the attacker and the victim. For instance, to form a timing channel using some cache-based attack, the victim and the attack processes must execute on the same processor core for a specific amount of time. The attacker's ability to adhere to these assumptions can significantly impact the capacity or sensitivity of the channel.

Over the last few years, researchers have demonstrated the feasibility of a wide range of timing-based covert and side channels. Szefer [4] presents a comprehensive overview of timing attacks that are feasible due to vulnerabilities in processor architecture. Execution time differences for various instructions, resource sharing, impact of functional unit's state on program execution (e.g., branch prediction) and timing behaviors of memory subsystems (e.g., cache and prefetcher, etc.) are some characteristics of modern processors that lead to microarchitectural timing channels.

2) Power Channel: In power SCA [56], [57], an attacker measures the switching power traces of an electronic component during operation and then employ mathematical analysis on the traces to extract secret information. Basic premise of such an attack lies in the fact that the transient power traces of a chip leak its internal switching patterns, thereby leaking data secrets (e.g., cryptographic key) through the switching behavior. Shrinking technology nodes and increasing power density have made it possible for attackers to carry out power SCAs with increasing degree of success.

Attackers have utilized a wide-variety of techniques to extract information. A simple visual inspection of the power signal information known as Simple Power Analysis (SPA) [56] is utilized when the internal implementation is known to the attacker. If the attacker has complete access to a device, he/she resorts to template matching attacks. Template attacks [5] consist of a profiling step and an attack step. The attacker has the freedom to collect many samples in the profiling phase as he/she fully controls the device. In the profiling step, the parameters of the design are learnt from a device and a profile of the device is created. This profile is applied as a template to other copies of the same device in the attack phase.

Differential Power Analysis (DPA) [56] relies on the principles of statistical hypothesis testing, where the attacker measures the power consumption traces of a target device over several time steps by feeding a large number of input vectors. The attacker then partitions the resulting power traces into subsets. The difference in the average values of these subsets reveal the presence or absence of information leakage in the design. In the absence of leakage, the difference in average values tends to be zero as the choice of assigning a trace to a subset is purely random and is uncorrelated with the power measurements. On the other hand, a statistically significant difference implies that there exists a correlation between the partitioning and trace measurements. Unlike SPA, DPA does not require any knowledge about the underlying implementation and can be carried out in highly noisy environments. Correlation Power Analysis (CPA) [57] relies on using statistical models to estimate the correlation between the secret and the power consumption of the device when the secret is being used for computation. A CPA typically relies on building a model of the device's dynamic power consumption. The activity factor α is modeled using Hamming distance (HD) or Hamming weight (HW). The change in the bits of the input that cause a change in α can be modelled by the HD between the initial input and the changed input values or the HW of an input in case of a software implementation (e.g., on smart card). This HD or HW model serves as a good approximation to estimate the power consumption of a device. During a CPA attack, the attacker guesses the value of the secret and obtains as many traces as possible for each guess of the secret.

ML algorithms have also been applied to both profiling-based and non-profiling-based SCAs. In profiling-based approaches, where attackers have access to an exact copy of the attacked hardware, a supervised ML model can be trained based on data points in different profiling traces [58]. In non-profiling based approaches, where attackers do not have access to a copy of the device, unsupervised ML algorithms such as clustering are applied to reveal the secret information [59].

The growth of cloud-based service providers like Amazon and Google has led to an increase in multiple users sharing the same hardware resource, such as a Field Programmable Gate Array (FPGA). In such multi-tenant operating environments, remote power attacks are becoming feasible when an untrusted party shares resources with a trusted one [6], [11]. The attacker can infer information regarding the trusted program executing in the same resource as the attacker by accessing the power delivery network. Furthermore, in cloud settings, new attacks are emerging where malicious power/current surge caused by an untrusted process can create denial of service attack in another process mapped to the same FPGA device [60].

3) Electronmagnetic and Photonic Channels: Unintentional Electronmagnetic (EM) radiation from electronic devices is a well-known concern for semiconductor vendors due to the possibility of interference with wireless communication channels and potential health risks to the end-users [24]. However, EM radiation during a security-critical process could also lead to vulnerabilities due to its potential to leak information regarding the operation. EM emission characteristics are largely device dependent; hence it is difficult to develop break-one-break-all scenario for the attackers. The effectiveness of small magnetic loop antennas in detecting EM emission from ICs has been evaluated in various studies [61]. The signals captured by magnetic loop antennas are digitized for the extraction of the secret. EM signal for information leakage can be observed in various ways. Visual inspection of the time-domain representation of EM signals is called simple EM analysis (SEMA) [62]. SEMA can be considered as the EM equivalent of the SPA. EM signal can be transformed to frequency domain to perform visual analysis of the spectrogram to reveal information. SEMA approach has been used to extract secret information from various cryptographic processes, including RSA, Elliptic Curve-based Diffie Hellman (ECDH) and Elliptic Curve based Digital Signature Algorithm (ECDSA) [63]. Algorithms like ECDH and ECDSA are suitable for mobile devices and Internet of Things (IoT) platforms where a malicious end-user with complete physical access can compromise the cryptographic process using SEMA approach.

Simple visual observation of EM signal may not be sufficient for revealing information from many applications. A more sophisticated attack vector called Differential EM Analysis (DEMA), a variant of DPA for EM is proposed [64]. However, DEMA requires a large number of EM traces of a given operation to extract the secret bits that are involved in the process by observing the variation in EM emission. With the alteration of signal or register states between logic high and low, energy dissipation in a CPU varies and consequently the EM emission is impacted [61]. Moreover, alteration of signal states in a CPU depends on the instructions and variables. Hence, observation of EM emission for a large number of operations is useful in retrieving the instructions being executed and intermediate states of different variables.

4) Fault Injection: Fault attacks form a potent class of SCAs wherein the attacker can subvert the execution of the hardware by deliberately injecting a fault. A well-placed fault attack could cause the system to reveal secret information, such as the key bits [65]. Fault attacks have also emerged as major threats for a program executed by a processor. For example, precisely flipping the status flags can allow an attacker to bypass the authentication process giving unauthorized control or privilege escalation [7]. These faults can be injected by causing a glitch in the underlying hardware. The attacker typically attempts to manipulate one or more of the devices' power supply or clock or utilizes a highly powerful laser to control the temperature of the device.

Fault attacks have been demonstrated on several crypto-functions such as Data Encryption Standard (DES), Advanced Encryption Standard (AES), International Data Encryption Algorithm (IDEA), Secure and Fast Encryption Routine (SAFER) and Blowfish. However, not all faults are exploitable. Hence, it requires careful profiling of the fault space to identify the set of exploitable faults. In AES, it has been demonstrated that a well-placed fault injected in between the seventh and ninth round operation could cause the device to reveal the entire key with as few as eight faulty ciphertexts.

More recently, attacks like PlunderVolt [31], VoltJockey [38] and CLKScrew [7] have demonstrated that fault attacks are not restricted to crypto-cores but can also impact general purpose SoCs. Both CLKScrew and Plundervolt are software generated fault attacks. The attacker leverages the access to clock or energy management APIs for injecting the fault. CLKScrew exploits the dynamic voltage frequency scaling utility to extract secrets from ARM Trust-Zone. PlunderVolt utilizes the power management utility to compromise the execution of Intel's SGX.

Apart from the above discussed side and covert channels, test and debug infrastructures usually provide privileged access to critical hardware resources such as machine state and configuration registers. Insecure test and debug ports are potential attack surfaces for launching powerful low-level attacks. In 2012, a military grade FPGA was reported to have a backdoor in the JTAG port, which allows the attacker to retrieve the AES key for decrypting the protected bitstream [66]. Rajput *et al.* [67] summarized the security attacks and protections for the commonly used JTAG port. Valea *et al.* [68] performed a more complete survey of security threats and countermeasures in different test standards.

C. IP Theft and Counterfeiting Threats

Modern SoC and IC designs usually involve different forms of IPs, e.g., register transfer level (RTL) design (soft IP), gate level netlist (firm IP) and physical layout (hard IP). The owner's IP is outsourced to trustworthy offshore design houses/foundries for SoC integration or IC fabrication to reduce design complexity, time to market pressure and manufacturing cost. This can lead to various IP security threats.

In IP counterfeiting, an attacker illegally imitates the original design, creates counterfeited versions of the IPs/ICs, and sells them in the brand name of a genuine supplier. In cloning attack, an adversary copies the original design and supplies cloned versions of original IPs/ICs under his/her own label. These attacks result in integration of fake IPs/ICs in the electronics systems used in critical applications such as military, healthcare and banking, etc. The fake designs not only sabotage the genuine vendor's reputation and revenue but also lead to large consequences: (i) affecting the reliability and performance of the critical systems; (ii) containing malicious or backdoor logic that cause leakage of confidential information or assist to override the critical systems [69].

In RE attack, an attacker back engineers the design in order to deduce the design structure or functionality. This can be done by RE its various design forms such as RTL, netlist, layout (GDS-II), mask or a manufactured IC [70]. RE attack allows the adversary to realize his/her intentions of inserting backdoors or Trojans into the design and also enables counterfeiting and IC overbuilding, thereby entails reassessing trust in electronics hardware [70], [71].

D. Hardware Trojan

1) Classical Digital Trojans: Early HTs typically use a single trigger signal to activate the Trojan under a rare event. The *Trust-HUB* benchmarks [72] employ such a simple trigger mechanism, which is very sensitive to switching probability analysis. The *De-Trust* [73] project provides some HT designs that use multiple discrete trigger signals so that each trigger signal will be able to switch normally. These HTs, when activated, will violate explicitly specified design behavior in the design specification.

A comprehensive list of Trojan taxonomies [74], [75], benchmark sets [72], [75] and lessons [27] of these classical HT research have been documented. In what follows, we will discuss some recent HT designs and attacks.

2) Exploitation of Don't Care Conditions: Fern *et al.* [8] leveraged external don't care conditions (i.e., unspecified functionality) for HT design. For example, the design output may be unspecified under certain "illegal" input conditions or when the output is not yet valid. Such don't care HTs can be hard to detect since they are out of the functional specification. A more recent work hides HT in the unspecified functionality in obfuscated hardware designs [13]. Based on the fact that the design functionality under incorrect obfuscation keys cannot be explicitly specified in order to protect the correct key, the IP designer has numerous flexibility in implementing the obfuscation logic, including inserting malicious circuitry.

Nahiyan *et al.* [76] proposed a HT design by adding malicious state to the finite state machine (FSM). The idea is to use unoccupied state encoding to insert a floating Trojan state. The FSM will never transit to the dangling Trojan state during normal operation. The Trojan can be activated using fault attack to force the FSM into the malicious state.

Hu *et al.* [34] leveraged satisfiability don't cares for HT insertion. The Trojan uses a pair of signals that will never reach a specific input combination (e.g., cannot be logical '0' simultaneously due to path correlation) under normal operation as triggers. Thus, the Trojan will never be triggered during normal run although each trigger signal is able to switch. Similarly, fault injection is used to force the trigger signals into a desired condition to activate the Trojan. Such Trojan has recently been demonstrated on a multi-tenant FPGA, where the attacker can remotely activate the Trojan by deploying power wasting circuitry to induce considerable fluctuations in the on-chip signal delays and, consequently, timing faults [77].

3) Analog Trojans: Researchers have also demonstrated how to create analog HTs through slight modifications of the design layout [78]. Becker *et al.* [79] and Kumar *et al.* [80] insert analog HTs by changing the dopant polarity or ratio of input to transistors to cause a short circuit. These dopant-level HTs can be hard to identify since they do not introduce additional transistors but only modify circuit parameter. Liu *et al.* [10] demonstrate an analog HT that leaks the AES key by slightly modulating the amplitude or frequency of wireless transmission without violating the protocol specification. The HT cannot be detected using routine testing methods since it does not change the design functionality. The A2 Trojan [9] is a small and stealthy malicious analog circuitry. It only adds a single capacitor that siphons charge from nearby wires as they transit. When the capacitor is fully charged, it drives a victim flip-flop to a desired value to perform malicious activities, e.g., elevating privilege. The Trojan will remain dormant if the capacitor resets through leakage current due to inactive switching activities in the charging wires. A more recent work exploits analog/mixed-signal circuits for hardware Trojans, whose trigger mechanism is deployed in the digital domain while the payload is transferred to the analog domain via the on-chip test infrastructure [81].

4) Trojans Induced Aging and Performance Degradation: In a nanoscale semiconductor device, physical occurrences such as hot-carrier injection, electromigration, time-driven dielectric breakdown and negative bias temperature instability (NBTI) lead to aging phenomenon [82]. These physical occurrences are the result of the restrained design margins and transistor scaling. Even a small change in the transistor parameter may significantly affect the device performance and reliability [82], [83]. Device aging may result in failure of semiconductor devices during critical operations. Heavy reliance of SoCs on third party IPs (3PIPs) raises the possibilities of aging attacks. A rouge 3PIP vendor may accelerate the device aging process by covertly making malicious modifications in the design of 3PIPs, with an aim of causing a premature failure of an electronic device within the warranty period [84].

One prevalent way of launching an accelerated aging attack is through NBTI stress. NBTI refers to the increase in threshold voltage of a P-type metal oxide semiconductor (PMOS) transistor over time due to the charges trapped under the gate area by the negative bias applied between its source and gate terminals [82]. As NBTI is heavily dependent on the dynamic operating condition of the device. Attackers can control the supply voltage, temperature and input signal probability to increase a device NBTI stress to accelerate its aging effect. An attacker may force the device into continuous stress even in standby mode, by modifying/adding malicious circuitry. To accelerate the aging process, the attacker can use selected input vectors to maximize the NBTI stress on the target devices.

This attack is demonstrated by Kachave *et al.* [84] on digital signal processor (DSP). In this attack model, an attacker continuously applies NBTI stress during the standby mode of the device to accelerate the aging by either hardware or software modifications. In the hardware-based attack, an attacker introduces some alterations in the DSP hardware such that a rare event (hidden Trojan) triggers the application of input vectors that maximize NBTI stress. In the cross-layer attack, an attacker builds a program that automatically applies the test vectors on the DSP circuit to create the greatest stress during the operational mode.

5) Trojans Insertion Through Malicious EDA Tool: HT threat arises primarily from untrusted design process and supply chain. EDA tools, as an important element in this untrusted environment, can also assist in Trojan attacks.

Krieg *et al.* [85] demonstrated an automated HT insertion technique through light-weight modification to an open source synthesis tool. The modified FPGA synthesis front-end deploys a special look-up table (LUT), whose simulated design behavior is totally correct. In a second attack phase, the malicious back-end identifies this LUT and changes its functionality when translating the design into bitstream, which acts as a Trojan trigger. The challenge in detecting such HT lies in the lack of bitstream verification tools. In their successive work, the differences in how the don't care 'X' appears in logic simulation and implementation are exploited to create a Trojan trigger. The trigger signal 'X' will be logic 0 during simulation and logic '1' in hardware implementation. Thus, the HT will remain inactive during the design phase and will be automatically activated upon configured onto the FPGA. Similarly, light-weight modification to the synthesis tool will facilitate automated insertion of such HTs.

Besides, several HTs target emerging computing technologies. In [13], a HT is hidden in the obfuscation logic intended for IP protection. In [6], a remote HT attack targeting multi-tenant FPGAs deployed in the cloud was demonstrated.

E. Vulnerabilities and Attacks on Deep Learning Networks

1) Adversarial Examples: AI has been promoting fast in the recent decade, thanks to various deep neural networks (DNNs), which learn high-level features from raw data to solve many challenging object recognition problems end-to-end with very high accuracy and without requiring human intervention. Similar to any other fast-advancing fields, the infiltration of deep learning models into safety and security critical applications such as self-driving cars and face recognition payment systems make them an interesting target of attack.

A well-known vulnerability has been exposed in a surprising way by the input of adversarial examples. It was initially demonstrated by Szegedy *et al.* [86] that small intentionally designed perturbations added to the original input image can create an optical illusion for the DNN classifier at the inference phase. Adversarial example generation algorithms, such as fast gradient sign method [87], universal perturbations [88] and Carlini and Wagner (C&W) attack [89], have succeeded in subverting the deep learning model output with high success rate. Hardware accelerator for the generation of adversarial examples has also been proposed to improve the attack efficiency [36]. The imperceptibility of the perturbation and generalization ability across models further aggravate the damage of such attacks. Recent research suggests that adversarial example attacks can also be applied in the physical world [90] and incorporated with cameras [91]. Adversarial examples work across different media and are recognized by Open AI Inc. as a concrete problem in AI safety. They shatter the confidence of DNN implementation robustness, and extend the DNN attack surface beyond the software boundary. Although conventional techniques such as laser beam interference, memory collision and rowhammer have been deployed as means to attack DNN hardware, they must be subtly and significantly devised to exploit the unique characteristics of DNN. The target asset and threat model of a DNN attack are in many ways different from those of the cryptosystem. DNN has the transferability, noise immunity and graceful degradation properties that are absent in many other domain-specific computing solutions. Effectiveness and efficiency of attacks on DNN are often data, model and application dependent. In general, data plays a more significant role than the model and the model plays a more significant role than parameter optimization in the inference.

2) Hardware-oriented Attacks: Artificial Intelligence of Things (AIoT) is the convergence of AI and IoT infrastructure. Placement of cognitive computing and AI processing at the IoT edges can benefit in terms of privacy maintenance, bandwidth reduction and responsiveness. As a core enabler of innovation, dedicated hardware accelerators for efficient on-device inference are increasingly used for edge AI deployment. Commercially available hardware accelerators for local AI inferencing include Intel Neural Compute Stick 2 (NCS2), Google Coral, Nvidia Jetson Nano and Xilinx edge AI IP core. This new wave of edge intelligence in the AIoT age invites new attack vectors, which are methodologically different from software-oriented DNN attacks like the previously described input of adversarial examples. This is because adversarial examples that assume any input pixel can be precisely altered to any arbitrary value may not achieve the same desired outcomes when they are presented to a DNN hardware accelerator.

Fault injection attacks such as laser injection [92], glitch disturbance [93], memory collision [94] and rowhammering [95] can impact circuit operations within the DNN and are potential threats to edge intelligence. Straightforward fault injection will cause denial of service, but it also alerts attention. For instance, overheating the DNN hardware will not only affect classification but also suspend the system. Immediate damage control may be triggered to limit the benefits that can be reaped from suck attacks. Existing fault attacks on DNN focus mainly on model weight manipulations [95], [96]. Falsifying model parameters such as saturating last layer's bias [32] to converge the output to one specific class regardless of inputs or constraining modification magnitude on the weights of all layers [96] can be used to create selective input misclassification. These simulated attacks assume that the data stored in memory can be precisely manipulated to arbitrary values through fault injection, which are not realistic for real-world DNN hardware accelerators. Moreover, manipulation and interpolation of model parameters tend to leave footprints in memory or create conspicuous output patterns. Such persistent fault induction in the weights are likely to be directly detected by model readback and bypassed by parameter reloading. Although practical fault injection techniques such as laser beam interference [92] and Rowhammer [95] are able to perturb the output of DNN algorithm running on general purpose hardware, the attacks can be mitigated by low-precision numeral representation, as suggested in [95], which happens to be a common practice of existing deep learning accelerator for edge applications.

HTs pose a real threat for outsourced DNN IC design, fabrication or testing activity or the use of 3PIPs within DNN hardware. Successfully embedded stealthy trigger and payload into the activation layer [97] or memory controller [98] can cause misclassification. Fortunately, hardware attacks on edge deep learning applications have so far been constrained to DNN hardware on small scale (10 categories) classification [92], [94] or are based on simulated instead of physically induced faults [96], [98] on larger network such as ImageNet [99] (1000 categories) classification. One exception is the most recently reported stealthy misclassification attack on deep learning accelerator for ImageNet applications in [100]. This attack induces temporal fault into intermediate results of convolutional layer by introducing infrequent instantaneous glitches into the clock signal. The temporary perturbated data will propagate to the inference stage but they will be overwritten by the correct data after each prediction, leaving no trace for detection.

3) Model Extraction Attacks: Model extraction attack [101] occurs when attackers attempt to replicate a pre-trained model. Because of the amount of costly training data collected, a superior deep learning model trained for a specific task is a precious IP that an enterprise can monetize as a commodity through third party offerings or leverage as a technology barrier to competitors of the market. Unlike cryptosystems, model confidentiality is assumed as a trained DNN is a pricey IP. For this reason, there is strong incentive for opponents to steal the model so as to build similar performance AI products or solutions at substantially reduced cost. Existing model extraction attacks can be broadly divided into two categories: query-based and implementation-based. Query-based model extraction attack mainly utilizes the input-output relationship of the target model to build a substitute model that has the similar functionality [101]. In the scenario of embedded devices, the internal model is exposed to the risk of being attacked by malicious users who have physical access to the device by observing the I/O dataflow [102]. These users can then train a new model with similar performance based on the I/O pairs, i.e., replicating the original model. Unlike crypto engine, where all computations can be completed fully on chip, edge implementations of DNN models, except a few tiny models, require some off-chip communications for each inference. Implementation-based model extraction attack exploits side channel leakage during model execution. Fine-grained information could be obtained by tracking cache misses, memory access pattern, power consumption and hardware performance counters [103], [104]. Algorithms such as DPA and CPA can be applied to extract the number of parameters in each layer, the value of each parameter, the total number of layers and the type of activation function. Optimization techniques on DNN hardware, such as zero weight pruning, can be utilized to reduce the complexity of reverse engineering [105]. The success of model extraction can enable further exploitation of the security weaknesses of deep learning, such as evading systems thereby forcing incorrect predictions and revealing additional information from the training data to leak sensitive and confidential information.

IV. Countermeasures

A. Hardware Security Primitives

True random number generator (TRNG) and physical unclonable function (PUF) are two important hardware-intrinsic security primitives that provide built-in instead of bolted-on defense against various emerging threats and vulnerabilities arising at different phases of the IC life cycle or device operation. Compared with TRNG, PUFs have been very well surveyed by many researchers in recent years. For TRNG, we exemplify typical CMOS circuit implementations from four different entropy sources. For PUFs, we focus on the feasibility of its integration with other non-device signatures. Such a unique provenance proof is promising in detecting imposer, tampering, spoofing and fabrication attacks that aim to gain unauthorized access to system, data or premises.

1) TRNG: A random number generator is a device or software that generates sequences of unpredictable numbers. The ancient ways of using dice roll or coin toss to harvest natural randomness are too slow to meet the demands of modern computing systems. A pseudorandom number generator (PRNG) is an algorithm or a mathematical formula that can be used to produce a sequence of random numbers with a sufficiently long but finite period from a seed state. PRNGs that are suitable for the cryptographic applications are called cryptographically secure pseudorandom number generators (CSPRNGs). CSPNGs are designed from cryptographic primitives or hard mathematical problems to pass the next-bit test such that the $(k+1)$-th bit of a sequence cannot be successfully predicted in polynomial time from the knowledge of the first k bits. CSPRNG should also be resilient to the "state compromise extensions" attack, which is an attack that makes use of some known internal states to predict future outputs or recover previous outputs. On the contrary, a TRNG is a hardware security primitive that yields unpredictable random numbers even if the internal design details are all known. With infinite period, it provides higher security property than CSPRNG. TRNG designs that originated from solid-state devices typically harvest their randomness from four sources, namely noise, jitter, metastability and chaos.

Thermal noise is a good source of randomness because it is frequency- and technology-independent [106]. The weak thermal noise needs to be boosted by a wide-bandwidth amplifier, which can consume significant silicon area and power. Matsumoto *et al.* [107] added a silicon nitride (SiN) layer in a standard CMOS process to amplify the thermal noise to a measurable level without the amplifier but the extra SiN mask is itself expensive. Recently, Bae *et al.* [108] proposed a high-speed TRNG by harvesting the thermal noise from the biasing circuit of a common-mode operating comparator and the sampling uncertainty of a Delay Flip Flop (DFF). The idea is illustrated in Fig. 1. Common-mode noise is generated by connecting both inputs of a comparator to the output of a beta-multiplier voltage reference. The thermal noises of the comparator and the biasing circuit are added up and amplified by the differential-to-single ended (D2S) amplifier. The amplified noise is fed into a slicer to generate a full swing output, which is then sampled by a 3 GHz clocked DFF. By combining thermal noise and sampling uncertainty of the asynchronous input, this TRNG has a very high throughput of 3 Gbps. Its power consumption is also very high, 5 mW excluding the power-hungry external high-speed clock generator.

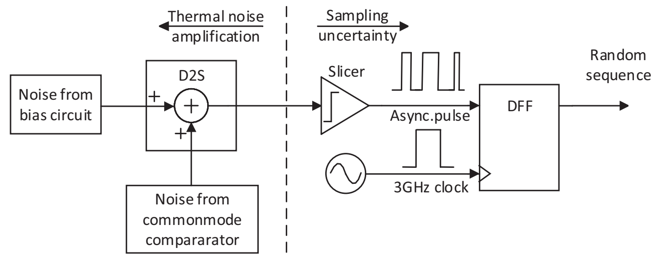


Fig. 1. Design concept of noise-based TRNG [108].

Conventional jitter-based TRNGs [109] use a slower jittery frequency clock to sample a faster clock. Using clock jitters of free running ring oscillators (ROs) as entropy source, the extractor design can be simplified, but additional power-hungry clock generators are required to provide adequate jitter variations. Yang *et al.* [110] proposed a process variation tolerant TRNG by exploiting the oscillation collapse in a double edge injected RO. To achieve the robustness against process variations, 32 stages with 8 selectable inverters per stage are used to provide the tuning space. Recently, a lightweight TRNG consisting of only two 9-stage current-starved ROs (CSROs) with an identical layout, a 3-stage regular RO and a 2-bit counter was proposed [111]. In order to maximize jitters and reduce power consumption, the inverters in the two CSROs are biased in the weak inversion region and the inverters in the regular RO are operating in the strong inversion region. Systemic biases in the beat frequency are effectively cancelled out by XORing the outputs of the two matched CSROs. The resulting random pulse width is used to clock gate the regular inverter-based RO to the 2-bit counter. This jitter-based TRNG, fabricated in a standard 65 nm, 1.2 V CMOS process, consumes only 260 μW at a bit rate of 52 Mbps and has a small footprint of 366 μm^2.

Metastability is a stable state of a dynamical system besides the system's state of least energy. Metastabilities in cross-coupled inverters, latches, DFFs and SRAMs [112] have been utilized to produce random bit streams at high bit rate, but complex post-processing units are usually required to eliminate the systematic bias. The key component of metasability-based TRNG of [112] is the metastability latch, which is designed based on a cross-coupled inverter pair with equal rise and fall time. A random bit is produced by the metastability latch in each cycle. To assure high entropy, a time-to-digital converter (TDC) is used to measure the settling time and tune the metastable latch against bias introduced by the process and temperature variations. The switching speed of the metastability latch cannot be too fast to prevent the settling time from exceeding the time resolution of the TDC. The latch size and load must also preserve the dominance of thermal noise over flicker noise. By combining three entropy sources of similar cross-coupled inverter pairs that share the same supply and clock, Intel [106] fabricated a fast TRNG in 14nm FinFET CMOS process that produces 3 full-entropy bits per clock cycle. The three bitstreams of at least 0.33 min-entropy/bit each are combined by a Barak-Impagliazzo-Wigderson (BIW) extractor [113]. Correlation suppressors and under-sampled feedback shift registers are used to de-correlate and whitening the raw data to generate 24 uncorrelated bits in every 64 clock cycles with an ultra-low energy consumption of 3 pJ/bit.

TRNGs can also be designed from chaotic system described by deterministic equations. At first sight, this may sound like God plays dice with complete law and order. Being extremely sensitive to the initial conditions, the disorder states of a chaotic system are very hard to be modeled mathematically even though they are produced by simple systems that obey precise rules. Chaos is, as described by the legendary Lorenz, "when the present determines the future, but the approximate present does not approximately determine the future." [114]. Chaos-based TRNGs [115] are typically designed by a chaotic map and a bit generation function. Unfortunately, the map characteristics are susceptible to process, voltage and temperature (PVT) variations. The optimal bit generation function for achieving the highest possible entropy rate from a map function is costly to implement, and consumes great power. An exceptionally energy-efficient implementation [116] is shown in Fig. 2. It consists of a 10-bit fine-SAR ADC, a 5-bit coarse-SAR ADC, a dynamic residue amplifier, and an XOR post-processing block. The ADC recursively amplifies the initial state of the system with environmental noise to produce a discrete time chaotic map. Due to quantization errors of the coarse-SAR ADC, the design is highly sensitive to the initial state. The switching power of fine-SAR ADC is reduced by using the coarse-SAR ADC to detect and skip switching. The design consumes only 82 nW of power and 0.3 pJ/bit of energy. A larger portion of the power savings are due to the dynamic residue amplifier and adaptive reset comparator.

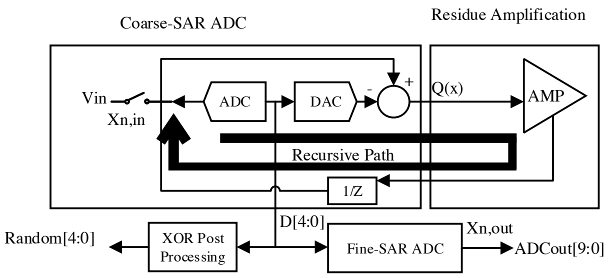


Fig. 2. Block diagram of chaos-based TRNG of [116]

As the need for publicly auditable randomness from applications like elections and lotteries increases, so is the demand for randomness beacon. A randomness beacon is a public server that produces completely unpredictable bit strings at regular intervals. During the Crypto Week last year, a new public randomness beacon called “League of Entropy” [117] was released by the American web-infrastructure and website-security titan company Cloudflare. Built upon the provably secure cryptographic architecture of drand [118], this is a network of beacons run by a consortium of global organizations and individual contributors to provide publicly verifiable, decentralized random outputs. Interestingly, Cloudflare actually sources her entropy from a video of a wall of lava lamps. These unpredictable visual data of floating blogs are converted to truly random numbers. Most recently, truly random numbers were also created from growing crystals [119].

2) PUF as Provenance Proof: PUF utilizes intrinsic manufacturing process variations to generate a unique unforgeable device fingerprint. A comprehensive review of PUFs can be found in [28], where different PUF structures, including the conventional delay-based or memory-based PUFs, and the emerging non-volatile memory (NVM) based PUFs, FinFET PUF, quantum secure PUF and sensor PUFs, have been surveyed. In the early stage of development, the reproducibility of PUF responses at different time and in different environmental conditions is the main practical issue that limits its industrial adoption. Majority voting, fuzzy extractor and reverse fuzzy extractor are three commonly used techniques to improve the reliability of a PUF. Majority voting votes for the most stable response by repeated application of the same challenge. It is a lightweight technique to enhance the reliability of a PUF at the expense of latency. Fuzzy extractor (FE) [28] increases the noise tolerance and uniformity of PUF response by error correction code (ECC) and hash function. As ECC decoding is too expensive for resource-constrained IoT devices, it is moved from the regeneration phase at the prover (device) side to the verifier (server) side by reverse fuzzy extractor (RFE) [120]. Instead of generating the helper data only once in the PUF enrolment phase, RFE generates helper data on site to different noisy versions of the same PUF response. While this eases the enrollment of strong PUF with a large number of CRPs, the disclosure of multiple helper data also increases the risk of side-channel information leakage. This problem can be mitigated by a well-structured PUF with balanced BER [121] or appending an Z-channel [122]. Today, PUFs have made their presence known in industry, e.g., Xilinx [123], NXP Semiconductor [124] and Qualcomm [125].

As a hardware root of trust, PUF has opened up new horizons for solving IoT security problems. The rise of IoT has created a huge influx of sensors and accelerated sensor standardization towards building a fully connected and cohesive supply chain. With sensors as the data feeder, a direct consequence is the new gloss on pushed media data and distinctively new interactions between human, events and devices. A promising new approach to assure real end point security against the imminent risk of sensor and data analytic attacks is to derive provenance proof from the unification of PUF responses and biometrics or other existing data analytic based security measures. This approach endows PUF systems with the capability to not only identifying the device, but also (i) authenticating the users who have privileged access to the device and its data, (ii) assuring the integrity of the data it generated or acquired, and (iii) responding actively to events occurred in the area of surveillance. Unlike conventional PUF designs, such interactive PUF systems are usually application- or sensor-specific, and have dedicated authentication protocols.

(i) Typical user and device authentication methods perform user and device authentication sequentially with a substantial message exchange. To protect the sensitive credentials during transmission, encryption keys are required to be stored in the end device, which are vulnerable to NVM key retrieval attacks [25]. In [126], [127], a completely different concept of unified user-device (UD) PUF was proposed to distinguish different users and devices by extracting raw biometric information like touch screen pressure or voice with the innate silicon sensor variations. The challenge to the UD-PUF in [126] is a series of binary coordinates that forms a pattern on the touchscreen. The response is a digital word obtained by quantizing the sequence of sensed pressure values read from an Android APP when the user traces the pattern. Unfortunately, the intrinsic parametric changes contributed by device fabrication process variations are not structurally harnessed, resulting in high identification error rate for the (same user, same challenge, different device) combination. The problem is intrigue as amplifying the parametric deviations to improve device identification will reduce the sensitivity of the user biometric whereas noise reduction in biometric information processing will distort device parametric distribution. Another “UD-PUF” was proposed in [128] for match-on-device applications. A strong PUF is required to generate an obfuscated biometric template by feeding the processed biometric feature into it. As small change in the challenge will cause a dramatic bit flips in the PUF response, the quantized biometric feature-based challenge has to be 100% accurate to ensure reproducibility of template in the authentication phase. This problem is mitigated by selecting the most robust biometric feature for each individual user using noise aware-interval optimized mapping bit allocation (NA-IOMBA). As

NA-IOMBA requires accurate noise samples/models over time for different conditions, the scheme can only generate one determinant template for a (user, device) combination. Once the template is leaked, the security of using the particular device will be compromised. This dilemma is resolved by a "UDhashing" scheme in [129]. UDhashing adopts a "fuse-on-device" and "match-on-server" strategy. Machine learning (ML) resilient strong PUF [130] is preferred to prevent the reuse of authentication credentials, and to achieve cancellable biometrics and system reconfigurability. To bind a device to its user, the user live biometric and device PUF response are unified by random projection into a bio-code at the end-device. The endpoint and the server are mutually authenticated by a zero-knowledge proof of the endpoint's secrets. The server is authenticated by the endpoint through the hashed PUF responses while the endpoint is authenticated by the server through the bio-codes. A correct biometric input of a user to his registered device and a correct response to a query from that device are both required to authenticate the bio-code. Neither the hashed PUF response nor the bio-code reveals the endpoint's secrets. The bio-code can be easily revoked, reissued or refreshed by a different challenge to prevent permanent compromise of the users' biometrics.

(ii) Similarly, PUF-assisted data-device authentication systems fill the gap of existing data and device independent authentication schemes in digital forensics. Digital images and videos have been increasingly exposed as important information or art carriers. Their easy-to-access and low-cost attributes also escalate image fraudulence. Two related problems are to be solved: detection of image tampering and authentication of the imaging device. Image tampering is typically detected by image watermarking [131], digital image forensics [132] and perceptual image hashing [133]. Of which perceptual image hashing is most effective in tamper detection. It is very sensitive to content-specific modifications and yet robust against normal content-preserving processing. Since such methods depend on a shared secret key for authentication, the security of the whole system will collapse if the secret key is compromised, lost or stolen. Source camera identification is mainly accomplished with ML based methods. By analyzing the structure and processing stages of the digital camera, appropriate features representing the unique device characteristics can be algorithmically extracted with the knowledge of lens aberration, sensor imperfection, color filter array interpolation and salient image features [134]. Existing works focusing on imaging device brand identification achieve very high accuracy but fail to distinguish individual devices from the same model and the same brand. Identifying individual camera devices have been increasingly studied in recent years based on photo response non-uniformity (PRNU) pattern [135], [136]. To achieve high reliability and accuracy, strict conditions in the acquisition process, number and content of training images as well as geometrical synchronization of testing images have to be met. More importantly, the same approach can also be used by a malicious user to extract the device features from publicly available images. To provide dual authentication without the aforementioned shortcomings, PUF-based data device authentication schemes have been proposed. PUF based perceptual image hash was first conceptualized in [137] for simultaneous tamper detection and source camera identification. This work shares the same PUF reliability problem as [128] since the data features were directly applied as the challenge to the underlying PUF. Alternatively, a data-device PUF (DD PUF) with relaxed reliability requirement was proposed in [138]. The method [138] imprints an indelible birthmark of the camera into its captured images for forgery detection. The robust data-device hash is produced by projecting the rotation-/scaling-invariant image features into the Bernoulli random matrix generated by the PUF responses. This hash is "keyless" and time-, data- and device-dependent. Attestation is non-repudiable as the perceptual image hash can only be generated by the timestamp of the image captured through the camera's tamper-resistant image sensor PUF. To achieve secure and accurate camera identification with reduced hardware overhead, the CMOS image sensor PUF [139] derived from fixed pattern noise of individual active pixel elements is utilized in both schemes [128], [138].

(iii) Existing PUFs, including the CMOS image sensor PUF [139], are typically triggered by server-provided challenges. Since the challenges are independent from the sensing targets, it is difficult to control the attestation frequency, resulting in either redundant or inadequate security tagging. Traditional frame-based imager generates too much redundant background data, which limits its processing bandwidth in high-speed and privacy-preserved video surveillance applications. Dynamic vision sensor (DVS), also known as neuromorphic vision sensor, provides a solution to design PUF system that is capable of responding actively to incidents occurred in the surveillance scene. DVS responds only to temporal intensity change and records only sparse asynchronous address-events with precise timing information. It has low latency, high dynamic range and significantly reduced data size. These features are exploited to make an event-driven PUF in [140]. It adds only three transistors per DVS pixel to harness the entropy from the fabrication process variability. The PUF response can only be triggered by and is uniquely dependent on the asynchronous addressed event detected in the scene without being interfered by the simultaneous firing of other address events. The package of address events acquired by the DVS camera is tagged by the event-driven PUF response using a keyed Hash-based Message Authentication Code (HMAC). This is believed to be the first event-driven PUF system to fill the forensic gap of simultaneously authenticating the event data integrity and source camera identity.

B. System and Architectural Protection Techniques

Resource sharing is inevitable as it leads to more efficient computation. Software processes share memories, datapaths, accelerators, monitors, sensors and I/O. Hardware IP cores require shared access to on-chip interconnect and memories. Yet, it is a challenge for security since any entity must consider information leakage through shared resource especially when computing on sensitive data. The cache side channel is a key example of this that has been exploited countless times for nefarious purposes. Resource isolation is a key security mechanism that is often difficult to implement in practice.

1) Trusted Execution Environment: One common approach for software isolation is a trusted execution environment (TEE). TEE uses hardware mechanisms to ensure that isolation properties are properly enforced. These properties, in general, enforce rules that provide a fixed set of resources for a sensitive computation, and assurances that those computations are hidden from other system users. There are a number of different TEEs. Intel's SGX [141] uses enclaves – a protected environment that contains the code and data of a security-sensitive computation. SGX performs isolation by setting aside a memory for trusted computation and isolating the memory from any other access including kernel, hypervisor and DMA accesses. ARM Trust-Zone [142] has two worlds. Sensitive computations are put into the secure world and are isolated from code running in the normal world.

2) Cache Side Channel Mitigations: Cache side channel mitigations attempt to minimize or eliminate information leakage by isolating secure and non-secure accesses to the cache. Cache partitioning is one class of approaches that attempts to separate the cache to avoid conflicts. Partitioning can be performed in various ways, including static locking (PLCache) [143], dynamic locking [144], page coloring [145], and selective cache flushing [146]. Randomization is another class of techniques where of cache access patterns are permuted to minimize any information leakage on conflicts, e.g., RPCache [143]. Other mitigations include DAWG [147], InvisiSpec [148], non-monopolizable caches [149], Intel Cache Allocation Technology [150], and CATalyst [151].

It is difficult to develop and implement a cache mitigation scheme. For example, Ardeshiricham *et al.* [152] showed that the well-known PLCache [143] mitigation was flawed, and developed a fix to the vulnerability that was formally verified to be secure. This points to the need for any mitigation to come with proof that they are correct. Property driven hardware security [20] advocates for such an approach where the threat model is formally specified as properties, e.g., SystemVerilog Assertion (SVA) assertions, information flow properties, etc., and hardware security verification tools provide assurance that the designs adhere to the specified properties.

3) Memory Protection: Many of the system and architectural threats revolve around performing proper access control on memory locations. This includes strict isolation of memory regions (e.g., non-secure processes should never read/write secure memory) and dynamic policies (e.g., a cryptographic key is written during secure boot process and is never accessed by anyone after that). Standard memory protections rely on a memory management unit (MMU). Common protections include access control through segmentation to provide isolation, data encryption to provide confidentiality [153], and hashing to provide integrity [154]. Protecting the memory access information, along with the confidentiality and integrity of the data, is also crucial. Oblivious RAM [155] is an example approach for access pattern protection. 3D integration is a powerful technique for hardware security and can be used for memory protection [156], e.g., embedded DRAM can mitigate threats related to off-chip data accesses. Anti-tamper techniques are also widely adopted by chip makers such as Altera [157], ON Semiconductor [158] and Cypress [159] to secure key storage. The most recent nonvolatile static RAM technology based anti-tamper memory [159] can provide a single or combined features of password protection, data destruction, functional destruction and physical destruction upon tampering.

4) Control Flow Integrity (CFI): Control flow integrity (CFI) defends against code reuse attacks by monitoring the program's flow of execution and attempts to ensure that it performs the correct sequence of operations. CFI is a general class of mitigation strategies that monitor and restrict the control flow decisions that a program makes. While there are many software CFI techniques, including some done in practice [160], there are fewer hardware based CFI techniques as they generally require substantial changes to the underlying microarchitecture. Hardware CFI defenses depend on a trusted hardware monitor integrated into the instruction pipeline or with access to the processor's debugging resources to analyze control flow information. de Clercq and Verbauwhede [161] classify CFI mitigation strategies into the followings: shadow call stack, labels, tables, finite state machine, branch regulation, instruction set randomization, signature modeling, and code pointer integrity. These mitigation strategies aim to monitor execution using a limited number of resources. Their differences are reflected in the resources that they monitor, how they track execution flows, and the type and amount of stateful information that must be stored.

C. Side Channel Protection Techniques

1) Timing-channel Countermeasures: Existing countermeasures against SCA explore both software and hardware-level approaches. A countermeasure could be detection of the attack at runtime or analysis of susceptibility during the design stage. It could also be a design approach (both software and hardware) for mitigating the covert or side-channel.

Most timing channels in cryptographic implementations occur due to the difference in execution time for different key and data inputs. As key and data inputs vary, the memory access pattern, branches, and various other operations become different across multiple executions that lead to leakage of information. Researchers have proposed constant-time techniques to eliminate such leakage. However, they are difficult to achieve through hardware-level re-implementation and may cause significant impact on performance [162]. Bitslicing technique has been explored to implement constant-time AES core with improved performance [163].

Researchers have developed compiler-based countermeasures to thwart timing channels. These techniques focus on introducing noise or randomization in the software implementation to eliminate timing leakage. Coppens *et al.* [164] proposed compiler-based automatic elimination of key-dependent control flow by removing conditional move instructions.

2) Power Side-channel Countermeasures: Power SCA countermeasures can be categorized as algorithmic, physical, or system-level. Algorithmic countermeasures insert additional operations that mask [165] or split [166] the sensitive computation. They have the advantage of being provably secure. Physical countermeasures rely on measurements for validating the security of the device. The problem of measuring the side-channel leakage of a device has been addressed in [39], [41],

[167], [168]. Works like [169], [170] use custom gates that consume power independent of the gate's switching. System-level countermeasures, such as [171]–[174], use the device's power supply to normalize or randomize the overall power consumption. While algorithmic and system-level countermeasures require additional circuitry, physical countermeasures use custom logic design methodologies to tackle the leakage. System-level countermeasures rely on injecting noise in the power supply, which reduces the signal-to-noise ratio in the side-channel leakage [171]–[174]. The specialized circuitry required by these schemes can drastically affect the area, power and performance of the design. For example, the popular algorithmic masking scheme [175] results in over $3\times$ of performance degradation. In the recent years, the need for incorporating security countermeasures in low-cost embedded hardware has motivated the emergence of efficient countermeasures like [176]–[178], where the algorithms available in the commercial EDA flows are leveraged to reduce area and delay overheads of these countermeasures.

3) EM Side-channel Countermeasure: Both hardware and software-level countermeasures have been proposed to thwart EM side-channel. Execution sequence randomization and randomization of LUTs have been explored as software-based methods [179], [180]. Certain pairs of instruction sequence may have distinguishing features in the EM signature that can be leveraged for the detection of security critical events [181]. Randomization of sequence can also be useful in mitigating such leakage. Accessing critical data using pointers may raise the difficulty of extracting access pattern information through EM analysis [182]. While masking of critical variables by random values during execution has been explored, they are found less effective in mitigating the leakage [183], [184].

Minimization of metal artifacts in the chip and use of Faraday cage packaging have been suggested as hardware-level countermeasures against EM emission [24]. Since EM emission is proportional to power consumption, low-power design methodology could also be useful. Introducing asynchronous design methods using multiple clocks may raise the difficulty of analyzing EM signature as different parts of the design will be switching at different frequency [24]. Analysis framework for evaluating hardware designs for EM side-channel vulnerability would be useful for early detection and integration of design-time mitigation techniques [185].

4) Fault Attack Countermeasures: Over the years, various countermeasures have been proposed for protecting digital designs against fault attacks. These countermeasures can be broadly classified as infective countermeasures and detection-based countermeasures. Detection-based countermeasures involve the addition of detection-circuitry such as parity or additional copies of the design in-order to detect the presence of a fault. Works such as [186]–[189] rely on parity based circuits while [190], [191] rely on redundant circuits. Works like [65], [192] attempt to thwart a fault attack by increasing the probability of unexploitable faults. They achieve this by transforming the fault space. Infective countermeasures such as [193], [194] on the other hand prevent the occurrence of a fault attack by making it impossible for the attacker to inject a fault. Works like [193] use diffusion-based technique where portions of the redundant and original outputs are swapped thereby making it harder for the attacker to identify exploitable faults. Another diffusion method is through the use of a fixed constant matrix to modify the output data [194]. Infection can also be achieved by adding dummy rounds in addition to the redundant datapath [195], [196].

However, a significant drawback of the above mentioned works is that they still require the design engineer to manually identify the vulnerable fault locations. This poses a significant challenge in larger designs. Thus, in recent years automatic identification of vulnerable locations has become an interesting area of research. With respect to fault attacks, the initial works were restricted to light-weight ciphers [197] or made strong assumptions such as restricting to bit-flip faults. Safari [198] can cater to a large class of block ciphers including add-rotate-xor (ARX) ciphers. It can comprehensively evaluate all possible fault scenarios, including those with multiple fault locations. Expfault [199] uses data mining to determine vulnerable components of a cipher. Solomon [200] is a formal verification based tool-flow that can map vulnerable regions in the specification to their corresponding gate-level or placed netlist representations. Feds [201] is a similar formal verification tool-flow that can map fault-attack vulnerable regions in the specification of a cipher to the corresponding lines in the source code for its implementation.

D. IP Protection Techniques

1) Hardware Watermarking: Hardware watermarking can be performed at electronic system level (ESL), high-level synthesis (HLS) level or logic synthesis level to protect an IP against threats such as piracy (or counterfeiting and cloning) and false claim of IP ownership. ESL or HLS based hardware watermarking is exemplified by binary encoding of author's signature in [202]. This technique embeds watermark in the pre-synthesis phase of HLS or behavioral synthesis in the form of additional design and timing constraints. The extra constraints encode the author's signature into a binary bitstream of ASCII characters. The high-level description of a design is converted into control data flow graph (CDFG). After scheduling the CDFG into control steps (CS), an interval graph (IG) is created wherein each node indicates a storage variable, and an edge between two nodes indicates the overlapping of the life time between two storage variables. Register allocation to these variables is performed by graph coloring. Each node is first assigned a unique number in increasing order of their lifetime. From the sorted list of nodes, each author signature bit is embedded as an extra edge in the IG by selecting a terminal node based on its node number. Bit '0' (or '1') is embedded by selecting an even (or odd) numbered terminal node. The extra constraints are thus imposed into the graph coloring problem for optimal register allocation. The strength of the authorship proof is assessed by the probabilities of coincidence (P_C) and tampering (P_T). P_C denotes the probability of coincidentally obtaining the same register allocation to the same storage variables as the signature by using any other register allocation methods. P_T denotes the probability of successfully corrupting the watermark by eliminating one or more signature bits by altering the color of a node.

Triple phase Watermarking [16] is another HLS watermarking scheme for protecting DSP hardware accelerators. A complex author-signature is formed by seven variables, 'γ', 'α', 'β', 'i', 'I', 'T' and '!', and embedded into three different phases of HLS. During the scheduling phase, on each occurrence of 'γ' digit, an operation in the non-critical path with the highest mobility is moved into the next immediate CS. During the resource allocation phase, in the odd CS on each occurrence of 'α' digit in the odd CS, hardware resources to the odd and even operations are reallocated to type 1 and type 2 venders, respectively; In the even CS, on each occurrence of 'β' digits , hardware resources to even and odd operations are reallocated to type 1 and type 2 vendors, respectively. During the register allocation phase, on each occurrence of 'i', 'I', 'T' and '!' digits, additional edges are added into the colored IG (CIG) to reallocate the registers to a set of storage variables. Specifically, 'i' is encoded as an edge between two prime nodes, 'I' an edge between two even nodes, 'T' i an edge between a pair of odd and even nodes, and '!' an edge between node number 0 and any other integer node.

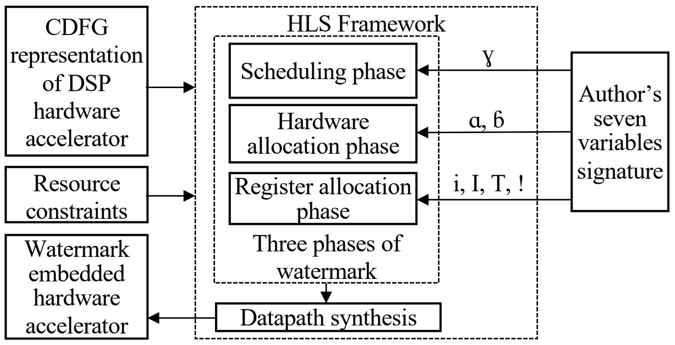

CDFG representation of DSP hardware accelerator
Resource constraints
Watermark embedded hardware accelerator
HLS Framework
Scheduling phase
Hardware allocation phase
Register allocation phase
Three phases of watermark
Datapath synthesis
γ
α, β
i, I, T, !
Author's seven variables signature

Fig. 3. Triple phase watermarking based IP protection technique.

Fig. 3 depicts the triple phase watermarking approach. Owing to the large number of signature variables embedded in three different phases of HLS, it is highly tamper-tolerant and has extremely low P_C. The complex signature combination in the three embedding phases has tightly constrained the solution space, making it highly improbable to find a design of the same functionality to also fulfill the additional watermark constraints by coincidence. By embedding the watermark at the highest level of design abstraction, the IP distributed at all lower levels of abstraction will also be protected without introducing integration complexity to the traditional design flow.

Besides HLS, Kirovski *et al.* [203] proposed the first logic synthesis watermarking method by implanting user and tool specific information into a combinational circuit through technology mapping. Design constraints are generated by hashing the owner signature using SHA-256 and a pseudo-random number generator. The watermark constraints are used to select the internal circuit nodes as pseudo-primary output to synthesize a new netlist with the minimum number of cells for a given technology library without changing the functionality of the original circuit. Instead of full technology mapping, Cui *et al.* [204] proposed an incremental technology mapping technique to adaptively synthesize part of the design for watermark insertion. Using a globally optimized master design, the slack sustainability of disjoint closed cones is assessed to determined their suitability as watermark hosts. The watermarked solution is generated by remapping only the selected closed cones according to the watermark constraint through incremental synthesis. As the closed cones are selected based on both slack and slack sustainability, the embedding capacity is maximized, and the watermark bits are stealthier than hosting them in non-critical paths determined merely by absolute timing slacks.

2) Hardware Steganography: A limitation of IP watermarking is that it is arduous, and not always possible, to optimize the design-dependent signature to increase its robustness without exceeding acceptable overhead. Hardware steganography is a promising alternative to watermarking. This is due to the following reasons: a) hardware steganography provides a seamless control to resolve ownership conflict and piracy detection; b) the secret stego-based hardware constraints are derived from the entropy threshold parameter instead of the combination and encoding process of signature variables. Consequently, the design overheads are reduced over IP watermarking. From design perspective, modeling the relationship between signature combination and design overhead to select a robust signature is extremely difficult. Hence, it is desirable to do away with signature dependency for IP protection.

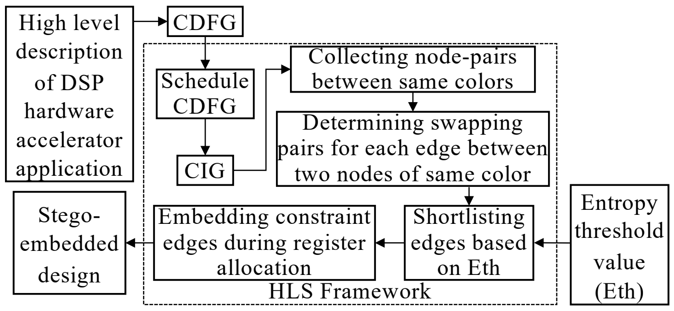

High level description of DSP hardware accelerator application
CDFG
Schedule CDFG
CIG
Collecting node-pairs between same colors
Determining swapping pairs for each edge between two nodes of same color
Shortlisting edges based on Eth
Embedding constraint edges during register allocation
HLS Framework
Stego-embedded design
Entropy threshold value (Eth)

Fig. 4. Entropy based hardware steganography approach.

In [17], a vendor signature-free entropy-based hardware steganography method is proposed to protect DSP cores. This approach is depicted in Fig. 4. The secret information is embedded in the register allocation phase of HLS through the CIG framework. The stego-constraints are derived from a set of edges between node-pairs of identical colors. To add an edge between two nodes of the same color, the color of one node in the pair needs to be swapped with another node in the CIG. There are a number of possible swapping pairs corresponding to each potential edge to be embedded. An entropy value is computed for each swapping pair as an indicator of the number of color transformations needed for the swapping. The entropy value for all swapping pairs of a potential edge is computed to determine its maximum entropy. Only those edges whose maximum entropy is less than or equal to a chosen entropy threshold (Eth) are qualified to be embedded as stego-constraints. The strength of the ownership proof is also measured by P_c. The difference is that a steganography technique is capable of embedding effectively a larger number of constraints than HLS based watermarking approaches. This is because it assumes no default constraint, while some constraints corresponding to the author's signature exist by default for watermarking approaches. All the stego-constraints corresponding to the chosen Eth are essentially embedded as the author's secret information. The amount of implanted stego-information and the strength of steganography can be increased by increasing Eth with negligible design overhead. Hence, this technique offers more designer control on the digital evidence implanted into the design.

To improve the robustness of steganography, two distinct phases viz. register allocation and functional unit (FU) vendor allocation of HLS are leveraged for stego-constraints insertion [205]. In addition, the author's stego-information generation involves cryptographic modules and stego-keys to enhance the protection against piracy and false ownership claim. The reasons are: (i) even if the secret constraints are compromised by an attacker, the owner has a meaningful and mathematical way to prove his constraints; (ii) the very large size stego-key (more than 600 bits and scalable with the size of the IP) is only known to the owner, and such a large key cannot be cracked by brute force; (iii) the stego-mark and ownership proof are strengthened by a stronger digital evidence by embedding the stego-information in two distinct phases of HLS.

The secret stego-constraints are generated using secret design data and stego-keys. The secret design data, obtained from CIG of target DSP application, are a set of elements where each element is represented by the indices (i, j) of a node pair (V_i, V_j) of identical colors in the CIG. A series of transformations involving row and column diffusions and cryptographic encryptions using the stego-keys are applied on the secret design data to obtain the stego-constraints in the form of a bitstream. Each bit in the stego-constraints is mapped to the hardware security constraints based on designer specific mapping rules. Further hardware security constraints are embedded during the register allocation and FU vendor allocation phases of HLS; thereby, generating a stego-embedded DSP design. This crypto-based dual phase steganography technique has also been applied along with structural obfuscation to be discussed in the next subsection to double the line of defense for securing JPEG compression-decompression hardware used in medical imaging systems [206].

3) Logic Obfuscation: Logic obfuscation is another effective hardware IP protection technique against illegal black-box reuse and RE. It inserts extra logic associated with dedicated obfuscation key inputs to functionally lock the design. Such design modification introduces programmability into the design such that the circuit functions properly only upon application of the correct obfuscation key and would otherwise malfunction.

Commonly used logic obfuscation techniques include XOR/XNOR and MUX based logic locking which can affect values of circuit internal nodes or the hardware information flow [207], [208]. Similarly, logic obfuscation can also be implemented by introducing programmable elements to withhold part of the logic for later configuration [209]. However, these obfuscation techniques are vulnerable to powerful functional oracle-guided SAT attacks that iteratively finds distinguishing input patterns (DIPs) to prune the wrong keys [210].

A number of anti-SAT logic locking techniques were proposed to increase the number SAT iterations required. For example, Xie et al. [18] leverage point function logic to reduce the number of wrong keys pruned by each DIP so that the number of DIPs required exhibits an exponential relation with the obfuscation key length. However, since such anti-SAT logic obfuscation approaches generally rely on AND-tree based point function structure, they suffer from removal attack and bypass attack [211], [212]. Several research work [213], [214] improved the obfuscation techniques to eliminate such vulnerability by adopting a corrupt-and-correct scheme which ensures that when the point function logic is removed, the circuit would not function properly. However, a recently proposed functionality analysis on logic locking (FALL) attack combines structural and functional analyses of the obfuscation circuit followed by a SAT-based key confirmation to successfully defeat such obfuscation techniques [215]. Another solution to developing anti-SAT logic locking techniques is to increase the time for each SAT iteration by using programmable logic and routing block networks to obfuscate the routing of selected wires as well as the logic of the gates preceding and succeeding the selected wires [216].

Another branch of logic encryption is the FSM based sequential obfuscation techniques. It involves augmenting the original FSM with additional states such that the FSM will start from a dummy state and can only reach a functional state upon receiving the correct key sequence [217]. Sequential obfuscation techniques can be breached if the FSM can be enumerated with its transition graph extracted [19].

E. Hardware Trojan Detection and Prevention Techniques

1) Pre-silicon Countermeasures: Pre-silicon HT detection techniques are designed to identify HTs in the early design phase. These techniques include *switching probability analysis*, *structural checking* and *security verification*.

Switching Probability Analysis based HT detection approaches are developed upon the assumption that the Trojan trigger signal should have extremely low switching probability in order to prevent the HT from being frequently activated. These methods try to identify the signals with switching activities significantly lower than the average through structural analysis or behavioral code analysis [218]–[220]. These research works have revealed the close connection between low controllability or observability signals and Trojan circuitry.

Structural Checking based HT detection methods attempt to extract structural features (e.g., gate type, gate count and manners of interconnections) specific to HT designs and perform detection leveraging techniques such as pattern matching [221]. They usually use a scoring algorithm to match such features against the circuit structures under test to identify Trojan circuitry. However, these methods may indicate false positives and suffer from scalability issues.

Security Verification can be used to detect certain types of HTs. It works by deriving formal security models for hardware designs and prove security properties such as *confidentiality* and *integrity* through formal approaches, e.g., SAT solving, model checking and type checking [222]–[224]. A security property violation indicates the existence of unintentional design flaw or intended malicious design modification. However, formal verification typically has the scalability problem and usually only works for detecting HTs at IP level.

2) Post-silicon Countermeasure: Pre-silicon HT detection methods check for malicious design modifications after chip fabrication. *Destructive RE*, which involves de-packaging and delayering the ICs and extracting the circuit structure from layout images, is a common approach for post-silicon HT detection. However, this costly and time-consuming process may fail when HT is only inserted into a small number of chips. On the contrary, non-destructive methods, including *functional testing* and *SCA*, are generally considered more viable in practice.

Functional testing aims to activate the rarely triggered Trojan circuitry and propagate the effect to an observable point. A major research vector is how to generate test vectors which can excite each rarely switching node in a circuit. Statistical approaches have provided one possible solution [225] while another attempt is guided tests against HTs in critical portions of the design [226].

SCA based methods detect HTs by analyzing physical IC parameters such as power consumption [227], path delay [228] and chip emissions [229], [230]. The major challenges with these methods lie in the lack of golden chip and the effect of process variation on side-channel measurements. To improve HT detection sensitivity, researchers exploit multiple side-channel parameters [231] or combine logic testing and SCA for directed pattern generation [232]. There are also efforts in developing "golden-free" solutions by estimating the golden signature through simulation [233] or comparing the signatures collected at different time windows [234].

3) Design-for-Trust (DFS) Techniques: DFS techniques insert dedicated logic to facilitate HT detection. Some techniques add dummy flip-flops and testing points to improve the controllability and observability of internal nodes to accelerate the Trojan activation process [235]. Another type of DFS techniques insert circuit infrastructure such as ring oscillators [236] and current sensors [237] to facilitate production screening and on-site monitoring of HT-infected chips or favor SCA-based HT detection. DFS techniques can also be applied to make HT insertion more difficult while detection easier as demonstrated in [238].

4) Runtime Monitoring Techniques: Due to the NP-completeness of several testing problems (e.g., controllability, observability and ATPG), it is impossible to guarantee that HTs can be completely eliminated before device deployment. Thus, it is desirable to employ runtime monitoring techniques to detect and prevent HT attacks in security critical systems. This can be done by monitoring critical signals [239], dynamic power [240] or EM radiation [241] and even through hardware-assisted formal approach [223].

5) 3PIP Trojan Detection: The majority of HT detection methods use Trojan benchmarks for evaluating their effectiveness. The task of detecting unknown HTs in 3PIP is more challenging due to the lack of knowledge about the Trojan implementation. Coverage of functional testing, locality of switching probability analysis and noise from process variation add difficulty to this process. Functional [242], [243] and security verification [224], [244]–[248] techniques are promising, with detection rate dependent on the quality of the properties. Unfortunately, specifying the right property for Trojan detection is a non-trivial task for 3PIP. Techniques for identifying HTs by matching the design structural [249] or control data flow [250] features to existing templates is reliable provided that the features of the Trojans are included in the feature library. New HTs that have not yet been reported may evade detection. Techniques that deploy on-chip monitors can still protect critical security assets from malicious activities triggered by unknown HTs [251], but full chip protection is infeasible due to the high design overheads.

6) HT Prevention Techniques: HT prevention techniques aim to make HT insertion more difficult or ideally impossible. *Logic obfuscation*, *split manufacturing* and *structural obfuscation* are three common Trojan prevention techniques.

Logic obfuscation techniques, initially proposed for IP protection, can also be leveraged as a HT prevention approach. Without the correct key, the functionality of the obfuscated circuit is locked to the untrusted foundry. This renders HT insertion a difficult task [252], [253].

Split Manufacturing can also help to prevent malicious design modification. It separates a design layout into Front End of Line (FEOL) and Back End of Line (BEOL) portions, which will be fabricated by trusted and untrusted foundries respectively. Without information about the BEOL portion, it is difficult for the untrusted foundry to embed a useful HT [254].

Structural Obfuscation transforms the design structure in order to hide its functionality and make the structure non-obvious/non-interpretable by an adversary. This renders RE harder, which thwarts malicious component/Trojan insertion. By considering the trade-off between design metrics such as area, delay and power during high-level transformations (HLTs), Lao and Parhi [255] obfuscated DSP circuits with huge structural alterations against RE and HT attacks without compromising their original functionality.

Hierarchical contiguous folding (HCF) is used to fold X cascaded stages to one hardware module, and N operations inside one stage to a hardware FU. All operations of one stage are performed before the next stage of operations. Different modes can be implemented by varying the number of stages in the cascaded structure. Some modes produce functionally invalid outputs but are otherwise meaningful from signal processing perspective. Other modes produce non-meaningful outputs. Manifold meaningful and non-meaningful modes are regulated through configured data. The functional mode of a DSP design is activated by applying a valid key to an FSM. If an invalid key or wrong configured data is applied, different modes will result in many equivalent circuits to obscure the DSP design structure.

Compiler based HLTs [256] is an alternative approach that targets mainly loop based DSP applications to achieve structural obfuscation. The exploitable HLTs include redundant operation elimination (ROE) by eliminating nodes in CDFG with matching inputs and operation type, logic transformation by altering some operation types in CDFG without changing the functionality, tree height transformation (THT) by parallelizing some sequential operations, loop unrolling by unrolling the loop body to reduce latency, and loop invariant code motion by moving non-iterative operations out of the loop. The aforementioned complier-based techniques considerably transform the CDFG and alter the RTL datapath of the DSP application post HLS. The latter alteration includes changes in the size and number of MUXes and DeMUXes, changes in the interconnectivity of FUs with MUXes and DeMUXes, change in the number of storage elements (registers), etc. By integrating particle swarm optimization based design space exploration (PSO-DSE) framework with the HLS process [256], the transformed/obfuscated graph can be scheduled with the optimal resource constraints, which minimizes the cost of the structurally obfuscated design. THT based [257] and hybrid transformations based [258] structural obfuscations are also applied to protect the JPEG codec hardware accelerators and fault secured DSP designs, respectively.

F. ML-assisted Solutions

Defenses against hardware security threats leveraging ML are mainly bifold:

1) ML for Detection: IC counterfeiting and HTs are two emerging threats to the IC manufacturing industry. In both cases, defective or malicious entities are injected as part of the system. Traditional inspection methods can be either very time-consuming or ineffective. As a result, ML models are used to automate the inspection procedure. Parametric measurements collected from on-chip sensors can be analyzed and classified using support vector machines (SVMs) to identify recycled ICs [259]. SVMs can also be utilized for real-time HT detection [14]. Additionally, various ML models such as SVM, random forest and multi-layer perceptron (MLP) have been applied to counter micro-architectural SCAs [15].

2) ML for Robust Architecture Design: Systems with robust designs stay ahead of security threats. Various designs attempt to combine ML and system characteristics. Yang *et al.* [102] leverage the memristor's obsolescence effect to design a secure neuromorphic computing system. Shan *et al.* [260] propose a machine learning assisted power compensation circuit that enhances the SCA-resistant capability with a smaller area and lower power overhead compared to traditional methods.

G. Countermeasures Against DNN Attacks

Most countermeasures against adversarial attacks on DNN can be dichotomized into proactive and reactive categories. The former intends to improve model robustness while the latter aims to detect adversarial inputs.

1) Proactive Measures: Proactive measures are carried out offline by three methods. *Adversarial training* retrains the model with off-the-shelf adversarial examples added into the original training dataset. Apart from the cost of crafting malicious images from known techniques, it is also limited by the assumption that the attacker is restricted to techniques that are known to the defender. *Gradient Masking* hides the gradient information from the adversaries. One example is to extract the probability vectors of a pre-trained model as soft labels to build a distilled model with the same architecture [261]. *Input Transformation* inhibits the adversarial effect by linear dimensionality reduction. In [262], principal component analysis is used to project the original data to the training data. Instead of building a fresh model specialized for the projected inputs, MagNet [263] reconstructs the inputs using autoencoders before is it trained with sufficient clean examples to move the tampered images towards the legitimate distribution. Hardware-oriented countermeasures have also been proposed to increase the robustness of DNN model. *Defensive Quantization* protects neural networks against adversarial attacks by controlling the Lipschitz constant of the network during quantization [264]. The hardware efficiency for small bitwidth data is still preserved.

2) Reactive Measures: Instead of passively regularizing model parameters in black box setting, reactive methods detect the adversarial inputs for follow-up actions. These methods can be divided into three main types: sample statistics, detector training and prediction inconsistency. *Sample Statistics* use features, such as density estimates calculated from the activations of the last hidden layer and Bayesian uncertainty extracted directly from the dropout layer, to detect illegitimate points lying far from and nearby the natural data manifold [265]. Nixon *et al.* [266] utilizes the sensor pattern noise (SPN) of the device for adversarial examples detection. The SPN Dash system, shown in Fig. 5, is introduced before the classification phase to detect adversarial perturbations after the image compression and submission stage. In Fig. 5, I_{enc}, SPN_{cur} and SPN_{dev} denote an input image after the image compression stage, the SPN of the current submitted I_{enc} and a reference SPN for a specific device, respectively. The main constraints for the generalization of this method are the susceptibility of SPN detector and device-dependent estimation accuracy. *Detector Training* augments the DNN subnetworks as adversarial input detector [267]. The additional module is trained by freezing the parameters of the original model to perform binary classification between the clean and adversarial inputs. This defense requires massive adversarial examples for training, and is prone to over-generalization of adversarial attacks. *Prediction Inconsistency* uses the degree of consensus among multiple models for the prediction of adversarial attacks. Wang *et al.* [268] integrates mutation and statistic hypothesis testing into the detection algorithm. As adversarial images are more sensitive to the model mutants than clean images, label change rate (LCR) is defined to assess the mutation sensitivity of DNN mutants. Statistic hypothesis testing is then applied to determine the cleanliness of the input based on its LCR. Cognizant of the greater freedom to adversarial abuse offered by the unnecessarily large feature space, Xu *et al.* [269] determines if an input is benign or adversarial by comparing the classification results of the initial and squeezed inputs against a predetermined threshold.

3) Other Hardware-oriented Measures: DNN models are typically implemented on GPU and application-specific accelerator platforms. The former has greater agility while the latter is more energy efficient. It is common to have the model training performed on GPUs with the inference executed in dedicated DNN accelerators. Thus, it is important to port

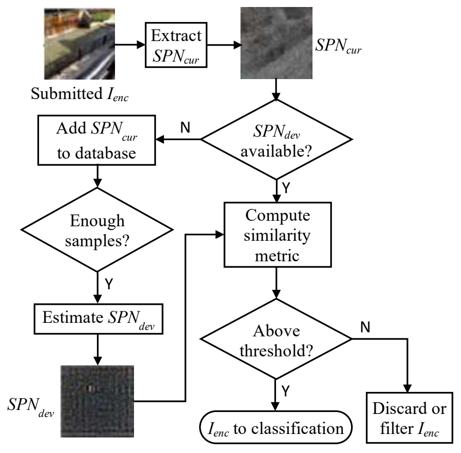


Fig. 5. SPN Dash framework [266].

defense algorithms into DNN accelerators to secure embedded intelligence. Most of the aforementioned countermeasures that target single precision floating-point arithmetic DNN implementation on GPU platform may not have considered the effects of fixed-point arithmetic and quantization on hardware accelerated DNN. Although truncated models have been used for the evaluation of the countermeasures proposed in [270] and [271], they are simulated under the GPU environment without physically integrated into the accelerator. Rouhani et al. [272] proposed the first end-to-end hardware accelerated detection framework that falls under the category of sample statistics. It exploits the possible adversarial sub-spaces spanned by the intermediate output feature maps through the Modular Robust Redundancy that architecturally mirrors the victim DNN model. An automated customization tool has also been developed for different resource-constrained platforms while maximizing the effectiveness of the defense. However, this defense was only tested on small scale classification. The complexity of the detection algorithm may increase substantially with more complex DNN models for large scale classification. To address this cost and performance trade-off, DNNGuard [273] is proposed. It is an elastic heterogeneous DNN accelerator architecture that enables simultaneous execution of the victim network and the detection network. It is also scalable to the implementation of various existing detection algorithms. Unfortunately, DNNGuard was also evaluated by simulation instead of physical implementation.

V. Hardware Security Tools

A. Security Verification Tools

Security verification tools for hardware/software systems have been surveyed in [30]. We summarize a few projects from the hardware side and also present some recent advances.

1) Academia Tools: One of the earliest hardware security verification tools was developed by the gate level information flow tracking (GLIFT) [274] project. It establishes the fundamental theories of hardware information flow tracking (IFT) by providing tracking logic formalization [275], [276] and complexity theories [277]. GLIFT has been employed to prove strong isolation in computer architecture [278], identify timing channel [47] and detect HTs [224]. The GLIFT project has evolved to higher level IFT methods such as register-transfer level IFT (RTLIFT) [279] due the verification performance bottlenecks at the gate level.

Similar to GLIFT, RTLIFT [279], Clepsydra [280] and VeriSketch [152] are a set of secure hardware design tools that employ fine-granularity security labels and label propagation policies to measure the flow of information but target RTL Verilog designs. RTLIFT has observed $\sim$5X improvement in verification performance as compared to GLIFT. Clepsydra provides a formal model for timing-only information flow and allows proving constant time properties in order to detect timing channels in caches and cryptographic cores. VeriSketch employs the sketch technique to automatically synthesize hardware designs that satisfy desired security properties such as confidentiality, integrity and constant time. Like GLIFT, the information flow security models developed by these projects are described in standard HDL while the security assertions are written in standard property specification languages, e.g., SVA. This allows hardware security verification to be performed under standard EDA verification environments.

The PCH-IP and VeriCoq projects provide several tools for verifying IP security and trust [245]–[247]. These tools define rules for converting RTL Verilog design to Coq[1] semantic circuit models. They use the Coq theorem prover to formally verify confidentiality properties on the Coq circuit models in order to detect malicious design modifications. VeriCoq [245] has recently been extended to the transistor level to verify the security of analog/mix-signal designs and detect analog HTs [281]. However, these projects tend to employ conservative rules to model information flow security behaviors, which can lead to false alarms in security verification.

SecVerilog [282] is an open source hardware security tool for proving timing non-interference [283] and eliminating timing channels in RTL designs. It incorporates a type system into Verilog and a timing label to verify information flow security at compile time. Timing non-interference is enforced by checking type rules. The SecVerilog tool has been extended to support mutable dependent types to solve the implicit downgrading problem [284] and the chisel HDL. SecChisel [285] can be used to create secure architectures, synthesize secure cryptographic accelerators and capture information leaks caused by hardware security flaws, timing channels and HTs through type checking.

2) Commercial Tools: EDA and hardware security companies have also released several secure hardware design tools.

Mentor Graphics SecureCheck is a security path verification tool running on top of the *Questa Formal* verification engine [286]. It uses assertion based formal verification to prove confidentiality and integrity properties in order to identify risky paths that will lead to security property violations.

JasperGold Security Path Verification [287] is a hardware security formal verification tool from *Cadence*. The tool runs on top of the *JasperGold Formal Verification Platform*. It employs sensitivity analysis to model the flow of information

[1]An iterative theorem prover named after its principal developer, Thierry Coquand.

in SoC designs and identify insecure design paths that can lead to tampering or leakage of critical information [288].

Prospect is a hardware security formal verification tool from *Tortuga Logic*. It uses GLIFT [274] to generate logic that tracks information flow through the circuit. That logic can then be analyzed by any functional verification tool to prove security properties [277]. The IFT logic is used solely for design time verification; no additional logic is added to the final circuit. *Radix-S* [289] and *Radix-M* [290] are another two hardware security tools from *Tortuga Logic*. *Radix-S* performs IFT based hardware security simulation while *Radix-M* performs hardware security emulation. Emulation allows for the verification of system properties across the entire SoC. Furthermore, it enables verification of properties that span across software, firmware, and hardware interactions, i.e., the "HardFails" [291].

Synopsys focuses more on reliability and functional safety verification. The *CustomSim* [292] is a tool set for device-level and interconnect reliability analysis, including infrared radiation drop, current density and electromigration, and device aging. The *VC Functional Safety Manager* [293] performs failure modes and effect analysis, unified fault campaigns management, annotation and calculation of metrics for the failure modes, effects and diagnostic analysis.

B. *Security Driven Hardware Design Tools*

There is a recent move towards developing security driven hardware design tools. DARPA has recently launched the *SSITH* and *AISS* projects. Both aim to develop secure hardware design tools that allow security to be evaluated along with traditional design parameters.

Urdahl *et al.* [294] propose a property-driven design flow. Abstract properties are specified from the system-level and refined along the design process in order to provide a formal relationship between an abstract system model and its concrete implementation at the RTL. In a property driven solution to hardware security [20], high-level security specification is translated to lower level security policy, property, assertion and constraints in order to allow security to be formally verified on more concrete design models. This is demonstrated by a property specific approach to information flow security verification [295]. Ma *et al.* [296] present a security-driven placement tool for EM side channel protection. The idea is to create an EM leakage model and use this model to guide data-dependent register reallocation. Takarabt *et al.* [297] propose a pre-silicon evaluation methodology and tool that allow security verification to be run side by side with functional verification. The tool identifies vulnerabilities and the precise line of code where the vulnerability lies with additional characterization such as severity. Recent advances are also witnessed in security-driven metrics, models and computer-aided design (CAD) flows that integrate logic encryption, split manufacturing and camouflaging for secure hardware design [298], [299].

In [21], it is argued that security should be taken as an architectural design constraint in addition to time, space and power. This motivates a security-aware design flow starting from the choice of security primitives, protocols and architecture. Knechtel *et al.* [22] provide a comprehensive analysis about the role of EDA on hardware security. They identify the challenges yet to be resolved in effective compilation of security assumptions and constraints across different levels of abstraction, modeling and evaluation of hardware security metrics and holistic synthesis of security countermeasures without causing side-effects.

VI. Potential Research Directions

A. *System and Architecture Security*

System designers are constantly trying to balance the delicate tradeoff between performance, power, and area. They must now add security as another optimization criteria! Unfortunately, measuring "security" is a challenging but crucial aspect of hardware design. Security metrics are essential for any sort of vulnerability analysis, threat mitigation, and security verification. An ideal metric provides a precise measure of the severity of the threat. Security is a multifaceted notion covering a wide range of threat models. Therefore, it is unlikely that one metric can cover all threat models. Thus, we need different metrics to understand the various threats. Metrics that can combine multiple different threat models and can model a high dimension space are valuable. Metrics that provide relative comparisons between different design options are also extremely useful in making architectural design decisions [41]. IFT is a powerful and one of the most popular metrics for hardware security verification, but it enforces binary properties (flow or no flow); Quantitative IFT helps provide some finer resolution for security threat modeling [300].

Debugging is another important but overlooked aspect of hardware security. This is particularly challenging at the system and architectural level due to the complexity of the design and the interactions with many disparate software and hardware components. When verification uncovers a security vulnerability, as it inevitably will do, designers require techniques to help localize the source of this vulnerability and suggestions on how to redesign the system to mitigate the vulnerability. This is particularly important for vulnerabilities that involve both hardware and software.

B. *IoT and Cyber-physical (CPS) Security*

Security of IoT and CPS are becoming increasingly important. This is due to: (1) these systems interact with the physical world, and hence security issues in these systems may lead to major safety concerns; and (2) these systems are designed and manufactured under tight cost and time constraints, which typically do not allow them to go through rigorous security design and verification process. Generally, these systems include a hardware layer that consists of sensors and actuators, electronic components for communication, control and information processing, and a software stack. The hardware layer serves as the root of trust for the entire system. Manufacturers of these systems often use commercial off-the-shelf (COTS) components for the hardware layer and many open-source software modules in the software stack due to the cost/time constraints. Moreover, these systems tend to incorporate 'smartness' through integration of various AI techniques that enable them to act autonomously as well as adapt to an operating environment.

Due to their distinctive properties, these systems will require significant re-thinking in how security solutions – both design and verification – can be effectively integrated into them. Specific research directions will include, (1) systematic integration of security in IoT/CPS that considers the varying requirements of the target applications (e.g., power, performance, cost) with focus on automatic design/verification tools that enables tradeoff between security and other design parameters; (2) security of the sensor/actuator subsystem; (3) security of the COTS components, since they come through an untrusted supply chain, thus being subject to counterfeiting and various tampering attacks; and (4) security of the AI and ML techniques employed in these systems. Additionally, there will be increasing need for quantifying the security of these systems through development of appropriate, easy-to-use and easy-to-understand metrics.

C. ML for Hardware Security and Security of ML

ML models can be used to launch or defend attacks against hardware entities. Current ML-assisted countermeasures rely mostly on preliminary models such as SVMs, possibly due to the scale of the problem and limited training data. As the attacks are continuously being developed, more complex ML models such as DNNs may emerge for the prowess in data processing, which in turn requires a large amount of training data. Therefore, unsupervised learning can be a potential key to the problem as labeled data are usually much more valuable than unlabeled raw data. Furthermore, more effort is needed to direct ML-assisted methodologies towards robust architecture design than anomaly detection, since the damage has already been done in the latter case.

Conversely, AI hardware are themselves vulnerable. Although practical constraints such as limited accessibility and custom hardware optimization can reduce the success rate of adversarial attacks, RE and potential exploitation of backdoor or flaws through deployed hardware accessibility and unreliable supply chains of IC design remain the valid threats. Existing detection-based countermeasures mostly focus on software-level using off-line analyses, which are often too late for remedy, especially in real-time safety-critical applications. More research effort is desired in built-in resilience against adversarial examples and protection of hardware DNN against theft of confidential trained model through queries or side channels without compromising efficiency and accuracy. Modern primitives for securing hardware, such as PUF, obfuscation and metering, may be embedded into the deep learning hardware to help monitor or control access to sensitive assets. However, without considering the intrinsic weaknesses of DNN implementation, the overhead and performance penalty may be unacceptable. It boils down to having a hardware-supported solution that takes unique attributes of DNNs into account at design time for system-level defense. Another challenge is the evaluation of the protection measures on large-scale datasets and complex models. More attention needs to be paid in the scalability of defense methodologies. Unfortunately, the opaque nature of deep learning has worsened the defender's situation. This calls for a learning paradigm shift from data-driven to knowledge-driven for explainability enhancement. Explainability is the ability to provide reasons for a specific decision derived by the AI. It can be evaluated by interpretability and completeness. Interpretability aims at describing inner operations in a simpler way while completeness measures the level of preciseness for an explanation. The dilemma is a highly interpretable system is usually weak in prediction whereas a precise description is often hardly understandable. The tremendous computations involved in deep learning processing have added resistivity to provide explanations for its decision. The core of processing explainability enhancement is therefore to reduce the operational complexity of the target DNN by for instance, constructing a saliency map to underline the most influential operations.

D. Security-driven EDA

State-of-the-art EDA flow takes functional correctness and performance budgets as primary design constraints. Incorporating security as an additional dimension of the hardware design space and enabling security properties to be evaluated along with traditional design parameters is a promising yet challenging research direction for both the hardware security and EDA communities. We need to develop more standardized hardware security models to allow security properties to be mapped and verified across different levels of abstractions. In addition, we need to derive effective security metrics to measure security as a quantifiable design variable.

VII. Conclusion

Hardware security involves multiple levels of abstraction in the computing system stack. In view of the enormously broad focus and attractivity of this field, it is not possible to comprehensively survey the voluminous publications, multidisciplinary and vast diversity of problems and solutions in one paper. In this paper, we surveyed and discussed the recent advances in selective sub-fields of hardware security. Specifically, we presented attacks and countermeasures on secure architectures, IP components and DNN models, as well as the design and niche applications of two popular hardware-intrinsic security primitives. We also outlined recent efforts in developing security-driven hardware design tools. Hardware attacks and countermeasures are rapidly evolving. It is not surprising that a different shortest bar of the wooden barrel can be identified with each major change in processor architectures and computing technologies. We believe that the rally between hardware attack and defense will remain a vibrant presence for a long time. It is therefore our aim that this review will alert the hardware designers and tool developers to pay additional attention to significant security gaps not addressable by traditional hardware design and verification methodologies.

References

[1] Bits, Please, "Extracting Qualcomm's keymaster keys - Breaking Android full disk encryption," 2016, [Online]. Available: http://bits-please.blogspot.com/2016/06/extracting-qualcomms-keymaster-keys.html.

[2] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," *ArXiv e-prints*, 2018.

[3] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown," *ArXiv e-prints*, 2018.

[4] J. Szefer, "Survey of microarchitectural side and covert channels, attacks, and defenses," *Journal of Hardware and Systems Security*, vol. 3, no. 3, pp. 219–234, 2019.

[5] O. Choudary and M. G. Kuhn, "Template attacks on different devices," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, Aug. 2014, pp. 179–198.

[6] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on FPGAs," in *Des. Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2018, pp. 1111–1116.

[7] A. Tang, S. Sethumadhavan, and S. Stolfo, "CLKSCREW: exposing the perils of security-oblivious energy management," in *USENIX Security Symposium (USENIX Security)*, Aug. 2017, pp. 1057–1074.

[8] N. Fern, S. Kulkarni, and K. T. T. Cheng, "Hardware Trojans hidden in RTL don't cares - automated insertion and prevention methodologies," in *IEEE Int. Test Conf. (ITC)*, Oct. 2015, pp. 1–8.

[9] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," in *IEEE Symp. on Sec. and Priv. (SP)*, May 2016, pp. 18–37.

[10] Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris, "Silicon demonstration of hardware Trojan design and detection in wireless cryptographic ICs," *IEEE Trans. VLSI Syst.*, vol. 25, no. 4, pp. 1506–1519, 2017.

[11] M. Zhao and G. E. Suh, "FPGA-based remote power side-channel attacks," in *IEEE Symp. on Sec. and Priv. (SP)*, May 2018, pp. 229–244.

[12] J. V. Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution," in *USENIX Security Symposium*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 991–1008.

[13] W. Hu, Y. Ma, X. Wang, and X. Wang, "Leveraging unspecified functionality in obfuscated hardware for Trojan and fault attacks," in *Asian Hardware Oriented Security and Trust Symposium*, Xi'an, China, Dec. 2019, pp. 1–6.

[14] A. Kulkarni, Y. Pino, and T. Mohsenin, "SVM-based real-time hardware Trojan detection for many-core platform," in *Int. Symp. on Quality Electronic Design*. IEEE, Mar. 2016, pp. 362–367.

[15] M. Alam, S. Bhattacharya, D. Mukhopadhyay, and S. Bhattacharya, "Performance counters to rescue: A machine learning based safeguard against micro-architectural side-channel-attacks." *IACR Cryptology ePrint Archive*, vol. 2017, p. 564, 2017.

[16] A. Sengupta, D. Roy, and S. P. Mohanty, "Triple-phase watermarking for reusable IP core protection during architecture synthesis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 4, pp. 742–755, 2018.

[17] A. Sengupta and M. Rathor, "IP core steganography for protecting DSP kernels used in CE systems," *IEEE Trans. Consum. Electron.*, vol. 65, no. 4, pp. 506–515, 2019.

[18] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating sat attack on logic locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 2, pp. 199–207, 2019.

[19] T. Meade, Z. Zhao, S. Zhang, D. Pan, and Y. Jin, "Revisit sequential logic obfuscation: Attacks and defenses," in *IEEE Symp. On Cir. and Syst. (ISCAS)*, 2017, pp. 1–4.

[20] W. Hu, A. Althoff, A. Ardeshiricham, and R. Kastner, "Towards property driven hardware security," in *International Workshop on Microprocessor and SOC Test and Verification (MTV)*, 2016, pp. 51–56.

[21] P. Ravi, Z. Najm, S. Bhasin, M. Khairallah, S. S. Gupta, and A. Chattopadhyay, "Security is an architectural design constraint," *Microprocessors and Microsystems*, vol. 68, pp. 17–27, 2019.

[22] J. Knechtel, E. B. Kavun, F. Regazzoni, A. Heuser, A. Chattopadhyay, D. Mukhopadhyay, S. Dey, Y. Fei, Y. Belenky, I. Levi, T. Güneysu, P. Schaumont, and I. Polian, "Towards secure composition of integrated circuits and electronic systems: On the role of EDA," in *Des. Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2020, pp. 508–513.

[23] Y. Lyu and P. Mishra, "A survey of side-channel attacks on caches and countermeasures," *Journal of Hardware and Systems Security*, vol. 2, no. 1, pp. 33–50, 2018.

[24] A. Sayakkara, N.-A. Le-Khac, and M. Scanlon, "A survey of electromagnetic side-channel attacks and discussion on their case-progressing potential for digital forensics," *Digital Investigation*, 2019.

[25] S. E. Quadir, J. Chen, D. Forte, N. Asadizanjani, S. Shahbazmohamadi, L. Wang, J. Chandy, and M. Tehranipoor, "A survey on chip to system reverse engineering," *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 1, 2016.

[26] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan attacks: Threat analysis and countermeasures," *Proc. IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

[27] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware Trojans: Lessons learned after one decade of research," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 1, 2016.

[28] C. H. Chang, Y. Zheng, and L. Zhang, "A retrospective and a look forward: Fifteen years of physical unclonable function advancement," *IEEE Circuits Syst. Mag.*, vol. 17, no. 3, pp. 32–62, 2017.

[29] A. Chakraborty, N. G. Jayasankaran, Y. Liu, J. Rajendran, O. Sinanoglu, A. Srivastava, Y. Xie, M. Yasin, and M. Zuzak, "Keynote: A disquisition on logic locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 1952–1972, 2020.

[30] O. Demir, W. Xiong, F. Zaghloul, and J. Szefer, "Survey of approaches for security verification of hardware/software systems," Cryptology ePrint Archive, Report 2016/846, 2016, https://eprint.iacr.org/2016/846.

[31] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens, "Plundervolt: Software-based fault injection attacks against intel SGX," in *IEEE Symp. on Sec. and Priv. (SP)*, May 2020.

[32] Y. Liu, L. Wei, B. Luo, and Q. Xu, "Fault injection attack on deep neural network," in *Int. Conf Comput.-Aided Des. (ICCAD)*, Nov. 2017, pp. 131–138.

[33] D. J. Bernstein, "Cache-timing attacks on AES," *VLSI Design IEEE Computer Society*, vol. 51, no. 2, pp. 218 – 221, 2005.

[34] W. Hu, L. Zhang, A. Ardeshiricham, J. Blackstone, B. Hou, Y. Tai, and R. Kastner, "Why you should care about don't cares: Exploiting internal don't care conditions for hardware Trojans," in *Int. Conf Comput.-Aided Des. (ICCAD)*, Nov. 2017, pp. 707–713.

[35] O. Mutlu and J. S. Kim, "Rowhammer: A retrospective," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, pp. 1–1, 2019.

[36] H. Guo, L. Peng, J. Zhang, F. Qi, and L. Duan, "Hardware accelerator for adversarial attacks on deep learning neural networks," in *International Green and Sustainable Computing Conference (IGSC)*. IEEE, Oct. 2019, pp. 1–8.

[37] S. Pinto and N. Santos, "Demystifying ARM TrustZone: A comprehensive survey," *ACM Computing Surveys*, vol. 51, no. 6, p. 130, 2019.

[38] P. Qiu, D. Wang, Y. Lyu, and G. Qu, "VoltJockey: Breaching TrustZone by software-controlled voltage manipulation over multi-core frequencies," in *ACM SIGSAC Conference on Computer and Communications Security*, Nov. 2019, pp. 195–209.

[39] J. Demme, R. Martin, A. Waksman, and S. Sethumadhavan, "Side-channel vulnerability factor: A metric for measuring information leakage," in *Int. Symp. on Computer Architecture (ISCA), Portland, OR, USA*. IEEE Computer Society, 2012, pp. 106–117.

[40] B. Mao, W. Hu, A. Althoff, J. Matai, Y. Tai, D. Mu, T. Sherwood, and R. Kastner, "Quantitative analysis of timing channel security in cryptographic hardware design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 9, pp. 1719–1732, 2017.

[41] A. Althoff, J. Blackstone, and R. Kastner, "Holistic power side-channel leakage assessment: Towards a robust multidimensional metric," in *Int. Conf Comput.-Aided Des. (ICCAD)*. IEEE, Nov. 2019, pp. 1–8.

[42] Y. Bulygin, J. Loucaides, A. Furtak, O. Bazhaniuk, and A. Matrosov, "Summary of attacks against BIOS and secure boot," *Defcon-22*, 2014.

[43] W. A. Arbaugh, D. J. Farber, and J. M. Smith, "A secure and reliable bootstrap architecture," in *IEEE Symp. on Sec. and Priv. (Cat. No.97CB36097)*, 1997, pp. 65–71.

[44] I. Lebedev, K. Hogan, and S. Devadas, "Secure boot and remote attestation in the sanctum processor," in *IEEE Computer Security Foundations Symposium (CSF)*. IEEE, 2018, pp. 46–60.

[45] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti, "A large-scale analysis of the security of embedded firmwares," in *USENIX Security Symposium (USENIX Security)*, 2014, pp. 95–110.

[46] P. Subramanyan, S. Malik, H. Khattri, A. Maiti, and J. Fung, "Verifying information flow properties of firmware using symbolic execution," in *Des. Autom. Test Europe Conf. Exhib. (DATE)*. IEEE, Mar. 2016, pp. 337–342.

[47] J. Oberg, S. Meiklejohn, T. Sherwood, and R. Kastner, "Leveraging gate-level properties to identify hardware timing channels," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 9, pp. 1288–1301, 2014.

[48] A. Basak, S. Bhunia, and S. Ray, "A flexible architecture for systematic implementation of SoC security policies," in *Int. Conf Comput.-Aided Des.* IEEE, 2015, pp. 536–543.

[49] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, "Lest we remember: cold-boot attacks on encryption keys," *Communications of the ACM*, vol. 52, no. 5, pp. 91–98, 2009.

[50] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," *ACM SIGARCH Comput. Archit. News*, vol. 42, no. 3, pp. 361–372, 2014.

[51] D. Brumley and D. Boneh, "Remote timing attacks are practical," *Computer Networks*, vol. 48, no. 5, pp. 701–716, 2005.

[52] D. Gullasch, E. Bangerter, and S. Krenn, "Cache games–bringing access-based cache attacks on AES to practice," in *IEEE Symp. on Sec. and Priv.* IEEE, 2011, pp. 490–505.

[53] R. Roemer, E. Buchanan, H. Shacham, and S. Savage, "Return-oriented programming: Systems, languages, and applications," *ACM Transactions on Information and System Security (TISSEC)*, vol. 15, no. 1, pp. 1–34, 2012.

[54] Z. Wang and R. B. Lee, "Covert and side channels due to processor architecture," in *Annual Computer Security Applications Conference (ACSAC)*. IEEE, 2006, pp. 473–482.

[55] C. Hunger, M. Kazdagli, A. Rawat, A. Dimakis, S. Vishwanath, and M. Tiwari, "Understanding contention-based channels and using them for defense," in *Int. Symp. on High Performance Computer Architecture (HPCA)*. IEEE, 2015, pp. 639–650.

[56] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO'99*, M. Wiener, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.

[57] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, M. Joye and J.-J. Quisquater, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, Aug. 2004, pp. 16–29.

[58] S. Picek, A. Heuser, A. Jovic, S. A. Ludwig, S. Guilley, D. Jakobovic, and N. Mentens, "Side-channel analysis and machine learning: A practical perspective," in *Int. Joint Conf. on Neural Networks*. IEEE, May 2017, pp. 4095–4102.

[59] J. Heyszl, A. Ibing, S. Mangard, F. De Santis, and G. Sigl, "Clustering algorithms for non-profiled single-execution attacks on exponentiations," in *Int. Conf. on Smart Card Research and Advanced Applications*. Springer, Nov. 2013, pp. 79–93.

[60] P. Koeberl, "Multi-tenant fpga security: Challenges and opportunities," in *Int. Symp. on Field-Programmable Gate Arrays*. New York, NY, USA: ACM, Feb. 2020, p. 23.

[61] E. Peeters, F.-X. Standaert, and J.-J. Quisquater, "Power and electromagnetic analysis: Improved model, consequences and comparisons," *Integration*, vol. 40, no. 1, pp. 52–60, 2007.

[62] N. Homma, T. Aoki, and A. Satoh, "Electromagnetic information leakage for side-channel analysis of cryptographic modules," in *IEEE Int. Symp. on Electromagnetic Compatibility*. IEEE, 2010, pp. 97–102.

[63] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, "ECDH key-extraction via low-bandwidth electromagnetic attacks on PCs," in *RSA Conference*. Springer, 2016, pp. 219–235.

[64] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2001, pp. 251–261.

[65] S. Patranabis, A. Chakraborty, P. H. Nguyen, and D. Mukhopadhyay, "A biased fault attack on the time redundancy countermeasure for AES," in *Revised Selected Papers of the 6th International Workshop on Constructive Side-Channel Analysis and Secure Design*, vol. 9064. Berlin, Heidelberg: Springer-Verlag, 2015, p. 189–203.

[66] S. Skorobogatov and C. Woods, *Breakthrough Silicon Scanning Discovers Backdoor in Military Chip*. Springer-Heidelberg, 2012, pp. 23–40.

[67] P. H. N. Rajput and M. Maniatakos, "Jtag: A multifaceted tool for cyber security," in *IEEE Int. Symp. on On-Line Testing and Robust System Design (IOLTS)*, Jul. 2019, pp. 155–158.

[68] E. Valea, M. Da Silva, G. Di Natale, M. Flottes, and B. Rouzeyre, "A survey on security threats and countermeasures in ieee test standards," *IEEE Design & Test*, vol. 36, no. 3, pp. 95–116, 2019.

[69] F. Koushanfar, S. Fazzari, C. McCants, W. Bryson, P. Song, M. Sale, and M. Potkonjak, "Can EDA combat the rise of electronic counterfeiting?" in *Design Automation Conference*, Jun. 2012, pp. 133–138.

[70] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 9, pp. 1411–1424, 2016.

[71] A. Sengupta, D. Kachave, and D. Roy, "Low cost functional obfuscation of reusable IP cores used in CE hardware through robust locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 4, pp. 604–616, 2019.

[72] A. Baumgarten, M. Steffen, M. Clausman, and J. Zambreno, "A case study in hardware Trojan design and implementation," *International Journal of Information Security*, vol. 10, no. 1, pp. 1–14, 2011.

[73] J. Zhang, F. Yuan, and Q. Xu, "DeTrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware Trojans," in *ACM SIGSAC Conference on Computer and Communications Security*, Nov. 2014, pp. 153–166.

[74] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware Trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[75] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, "Benchmarking of hardware Trojans and maliciously affected circuits," *Journal of Hardware and Systems Security*, pp. 85–102, 2017.

[76] A. Nahiyan, K. Xiao, K. Yang, Y. Jin, D. Forte, and M. Tehranipoor, "AVFSM: A framework for identifying and mitigating vulnerabilities in FSMs," in *Des. Autom. Conf. (DAC)*, Jun. 2016, pp. 89:1–6.

[77] D. Mahmoud, W. Hu, and M. Stojilovic, "X-attack: Remote activation of satisfiability don't-care hardware trojans on shared fpgas," in *Int. Conf. on Field-Programmable Logic and Applications (FPL)*, Aug. 2020, p. 8. [Online]. Available: http://infoscience.epfl.ch/record/278020

[78] A. Antonopoulos, C. Kapatsori, and Y. Makris, *Hardware Trojans in Analog, Mixed-Signal, and RF ICs*. Cham: Springer International Publishing, 2018, pp. 101–123.

[79] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy dopant-level hardware Trojans," in *Cryptographic Hardware and Embedded Systems - CHES 2013*, G. Bertoni and J.-S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 197–214.

[80] R. Kumar, P. Jovanovic, W. Burleson, and I. Polian, "Parametric Trojans for fault-injection attacks on cryptographic hardware," in *Workshop on Fault Diagnosis and Tolerance in Cryptography*, Sept. 2014, pp. 18–28.

[81] M. Elshamy, G. Di Natale, A. Pavlidis, M. Louërat, and H. Stratigopoulos, "Hardware Trojan attacks in analog/mixed-signal ICs via the test access mechanism," in *IEEE European Test Symposium (ETS)*, May 2020, pp. 1–6.

[82] B. Kaczer, T. Grasser, P. J. Roussel, J. Franco, R. Degraeve, L. . Ragnarsson, E. Simoen, G. Groeseneken, and H. Reisinger, "Origin of NBTI variability in deeply scaled pFETs," in *IEEE International Reliability Physics Symposium*, May 2010, pp. 26–32.

[83] S. Mahapatra, N. Goel, S. Desai, S. Gupta, B. Jose, S. Mukhopadhyay, K. Joshi, A. Jain, A. E. Islam, and M. A. Alam, "A comparative study of different physics-based NBTI models," *IEEE Trans. Electron Devices*, vol. 60, no. 3, pp. 901–916, 2013.

[84] D. Kachave, A. Sengupta, S. Neema, and P. Sri Harsha, "Effect of NBTI stress on DSP cores used in CE devices: threat model and performance estimation," *IET Computers Digital Techniques*, vol. 12, no. 6, pp. 268–278, 2018.

[85] C. Krieg, C. Wolf, and A. Jantsch, "Malicious LUT: A stealthy FPGA Trojan injected and triggered by the design flow," in *Int. Conf Comput.-Aided Des. (ICCAD)*. New York, NY, USA: ACM, Nov. 2016.

[86] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013.

[87] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[88] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 86–94.

[89] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symp. on Sec. and Priv. (SP)*, 2017, pp. 39–57.

[90] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[91] J. Li, F. Schmidt, and Z. Kolter, "Adversarial camera stickers: A physical camera-based attack on deep learning systems," in *Int. Conf. on Machine Learning*, Jan. 2019, pp. 3896–3904.

[92] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu, "Practical fault attack on deep neural networks," in *ACM Conf. on Comp. and Comm. Sec. (CCS)*. New York, NY, USA: ACM, 2018, pp. 2204—-2206.

[93] M. Agoyan, J.-M. Dutertre, D. Naccache, B. Robisson, and A. Tria, "When clocks fail: On critical paths and clock faults," in *Gollmann D., Lanet JL., Iguchi-Cartigny J. (Eds.), Smart Card Research and Advanced Application (CARDIS 2010)*, D. Gollmann, J.-L. Lanet, and J. Iguchi-Cartigny, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 182–193.

[94] M. M. Alam, S. Tajik, F. Ganji, M. Tehranipoor, and D. Forte, "RAM-Jam: Remote temperature and voltage fault attack on FPGAs using memory collisions," in *IEEE Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2019, pp. 48–55.

[95] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitraş, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *USENIX Conference on Security Symposium*. USA: USENIX Association, 2019, pp. 497––514.

[96] P. Zhao, S. Wang, C. Gongye, Y. Wang, Y. Fei, and X. Lin, "Fault sneaking attack: A stealthy framework for misleading deep neural networks," in *Des. Autom. Conf. (DAC)*. New York, NY, USA: ACM, Jun. 2019.

[97] J. Clements and Y. Lao, "Hardware Trojan design on neural networks," in *IEEE Symp. On Cir. and Syst. (ISCAS)*, 2019, pp. 1–5.

[98] Y. Zhao, X. Hu, S. Li, J. Ye, L. Deng, Y. Ji, J. Xu, D. Wu, and Y. Xie, "Memory Trojan attack on neural network accelerators," in *Des. Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2019, pp. 1415–1420.

[99] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.

[100] W. Liu, C. H. Chang, F. Zhang, and X. Lou, "Imperceptible misclassification attack on deep learning accelerator by glitch injection," in *Des. Autom. Conf. (DAC)*, Jul. 2020.

[101] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *USENIX Security Symposium*, Aug. 2016, pp. 601–618.

[102] C. Yang, B. Liu, H. Li, Y. Chen, W. Wen, M. Barnell, Q. Wu, and J. Rajendran, "Security of neuromorphic computing: thwarting learning attacks using memristor's obsolescence effect," in *Int. Conf Comput.-Aided Des.*, Nov. 2016, pp. 1–6.

[103] M. Yan, C. W. Fletcher, and J. Torrellas, "Cache telepathy: Leveraging shared resource attacks to learn DNN architectures," in *USENIX Security Symposium*. Boston, MA: USENIX Association, 2020. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/yan

[104] M. Alam and D. Mukhopadhyay, "How secure are deep learning algorithms from side-channel based reverse engineering?" in *Des. Autom. Conf. (DAC)*, Jun. 2019, pp. 1–2.

[105] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *Des. Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2018, pp. 4:1–4:6.

[106] S. K. Mathew, D. Johnston, S. Satpathy, V. Suresh, P. Newman, M. A. Anders, H. Kaul, A. Agarwal, S. K. Hsu, G. Chen, and R. K. Krishnamurthy, "μRNG: A 300–950 mV, 323 Gbps/w all-digital full-entropy true random number generator in 14 nm FinFET CMOS," *IEEE J. Solid-State Circuits*, vol. 51, no. 7, pp. 1695–1704, 2016.

[107] M. Matsumoto, S. Yasuda, R. Ohba, K. Ikegami, T. Tanamoto, and S. Fujita, "1200mum^2 physical random-number generators based on SiN MOSFET for secure smart-card application," in *IEEE International Solid-State Circuits Conference (ISSC)*, 2008, pp. 414–624.

[108] S. Bae, Y. Kim, Y. Park, and C. Kim, "3-Gb/s high-speed true random number generator using common-mode operating comparator and sampling uncertainty of d flip-flop," *IEEE J. Solid-State Circuits*, vol. 52, no. 2, pp. 605–610, 2017.

[109] Q. Tang, B. Kim, Y. Lao, K. K. Parhi, and C. H. Kim, "True random number generator circuits based on single- and multi-phase beat frequency detection," in *IEEE Custom Integrated Circuits Conference (CICC)*, 2014, pp. 1–4.

[110] K. Yang, D. Blaauw, and D. Sylvester, "An all-digital edge racing true random number generator robust against PVT variations," *IEEE J. Solid-State Circuits*, vol. 51, no. 4, pp. 1022–1031, 2016.

[111] Y. Cao, C. H. Chang, Y. Zheng, and X. Zhao, "An energy-efficient true random number generator based on current starved ring oscillators," in *Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, 2017, pp. 37–42.

[112] C. Tokunaga, D. Blaauw, and T. Mudge, "True random number generator with a metastability-based quality control," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 78–85, 2008.

[113] S. K. Satpathy, S. K. Mathew, R. Kumar, V. Suresh, M. A. Anders, H. Kaul, A. Agarwal, S. Hsu, R. K. Krishnamurthy, and V. De, "An all-digital unified physically unclonable function and true random number generator featuring self-calibrating hierarchical Von Neumann extraction in 14-nm tri-gate CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 4, pp. 1074–1085, 2019.

[114] T. Durden, "The approximate present does not approximately determine the future," 2013. [Online]. Available: https://www.zerohedge.com/news/2013-05-21/approximate-present-does-not-approximately-determine-future

[115] F. Pareschi, G. Setti, and R. Rovatti, "Implementation and testing of high-speed CMOS true random number generators based on chaotic systems," *IEEE Trans. Circuits Syst. I*, vol. 57, no. 12, pp. 3124–3137, 2010.

[116] M. Kim, U. Ha, K. J. Lee, Y. Lee, and H. Yoo, "A 82-nW chaotic map true random number generator based on a sub-ranging SAR ADC," *IEEE J. Solid-State Circuits*, vol. 52, no. 7, pp. 1953–1965, 2017.

[117] Cloudflare, "League of entropy." [Online]. Available: https://www.cloudflare.com/leagueofentropy/

[118] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford, "Scalable bias-resistant distributed randomness," in *IEEE Symp. on Sec. and Priv. (SP)*, 2017, pp. 444–460.

[119] E. C. Lee, J. M. Parrilla-Gutierrez, A. Henson, E. K. Brechin, and L. Cronin, "A crystallization robot for generating true random numbers based on stochastic chemical processes," *Matter*, vol. 2, no. 3, pp. 649 – 657, 2020.

[120] A. V. Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs," in *Keromytis A.D. (Eds.) Financial Cryptography and Data Security (FC 2012), Lecture Notes in Computer Science*, vol. LNCS 7397, Springer, Berlin, Heidelberg, 2012, pp. 374–389.

[121] J. Delvaux, "Security analysis of PUF-based key generation and entity authentication," Ph.D. dissertation, School of Cyber Science and Engineering, Shanghai Jiaotong University and Faculty of Engineering Science, KU Leuven, 2017, [Online]. Available: https://www.esat.kuleuven.be/cosic/publications/thesis-290.pdf.

[122] A. Schaller, T. Stanko, B. Škorić, and S. Katzenbeisser, "Eliminating leakage in reverse fuzzy extractors," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 4, pp. 954–964, 2018.

[123] E. Peterson, "Developing Tamper-Resistant Designs with Zynq UltraScale+ Devices," 2018, [Online]. Available: https://www.xilinx.com/support/documentation/application_notes/xapp1098-tamper-resist-designs.pdf.

[124] NXP Semiconductors Netherlands B.V., "NXP Delivers Enhanced Security Solution to Protect Personal Data for Payment and eGovernment Services," 2016, [Online]. Available: https://media.nxp.com/news-releases/news-release-details/nxp-delivers-enhanced-security-solution-protect-personal-data.

[125] X. Guo, D. M. Jacobson, Y. Yang, A. J. Drew, and B. M. Rosenberg, "Applying circuit delay-based physically unclonable functions (PUFs) for masking operation of memory-based PUFs to resist invasive and clone attacks," 2017, uS Patent 9787480B2, [Online]. Available: https://patents.google.com/patent/US9787480B2/en.

[126] R. A. Scheel and A. Tyagi, "Characterizing composite user-device touchscreen physical unclonable functions (PUFs) for mobile device authentication," in *International Workshop on Trustworthy Embedded Devices*. New York, NY, USA: ACM, 2015, pp. 3—-13.

[127] Y. Guo and A. Tyagi, "Voice-based user-device physical unclonable functions for mobile device authentication," *J. Hardware Syst. Security*, vol. 1, pp. 18—-37, 2017.

[128] N. Karimian, Z. Guo, F. Tehranipoor, D. Woodard, M. Tehranipoor, and D. Forte, "Secure and reliable biometric access control for resource-constrained systems and IoT," 2018.

[129] Y. Zheng, Y. Cao, and C. H. Chang, "UDhashing: Physical unclonable function-based user-device hash for endpoint authentication," *IEEE Trans. Ind. Electron.*, vol. 66, no. 12, pp. 9559–9570, 2019.

[130] Y. Cao, C. Liu, and C. H. Chang, "A low power diode-clamped inverter based strong physical unclonable function for secure and robust IoT authentication," *IEEE Trans. Circuits Syst. I*, vol. 65, no. 11, pp. 3864–3873, 2018.

[131] X. Liu, C. Lin, and S. Yuan, "Blind dual watermarking for color images' authentication and copyright protection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 5, pp. 1047–1055, 2018.

[132] L. Wen, H. Qi, and S. Lyu, "Contrast enhancement estimation for digital image forensics," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 14, no. 2, 2018.

[133] Z. Tang, X. Zhang, X. Li, and S. Zhang, "Robust image hashing with ring partition and invariant vector distance," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 1, pp. 200–214, 2016.

[134] B. Xu, X. Wang, X. Zhou, J. Xi, and S. Wang, "Source camera identification from image texture features," *Neurocomputing*, vol. 207, pp. 131 – 140, 2016.

[135] D. Valsesia, G. Coluccia, T. Bianchi, and E. Magli, "User authentication via PRNU-based physical unclonable functions," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1941–1956, 2017.

[136] A. Lawgaly and F. Khelifi, "Sensor pattern noise estimation based on improved locally adaptive DCT filtering and weighted averaging for source camera identification and verification," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 2, pp. 392–404, 2017.

[137] Y. Cao, L. Zhang, and C. H. Chang, "Using image sensor PUF as root of trust for birthmarking of perceptual image hash," in *IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*, 2016, pp. 1–6.

[138] Y. Zheng, Y. Cao, and C. H. Chang, "A PUF-based data-device hash for tampered image detection and source camera identification," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 620–634, 2020.

[139] Y. Cao, L. Zhang, S. S. Zalivaka, C. H. Chang, and S. Chen, "CMOS image sensor based physical unclonable function for coherent sensor-level authentication," *IEEE Trans. Circuits Syst. I*, vol. 62, no. 11, pp. 2629–2640, 2015.

[140] Y. Zheng, X. Zhao, T. Sato, Y. Cao, and C. H. Chang, "Ed-PUF: Event-driven physical unclonable function for camera authentication in reactive monitoring system," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2824–2839, 2020.

[141] F. McKeen, I. Alexandrovich, I. Anati, D. Caspi, S. Johnson, R. Leslie-Hurd, and C. Rozas, "Intel® software guard extensions (intel® SGX) support for dynamic memory management inside an enclave," in *Hardware and Architectural Support for Security and Privacy*, 2016, pp. 1–9.

[142] T. Alves and D. Felton, "TrustZone: Integrated hardware and software security-enabling trusted computing in embedded systems," 2014.

[143] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," in *ACM SIGARCH Comp. Arch. News*, vol. 35, no. 2. ACM, 2007, pp. 494–505.

[144] D. Page, "Partitioned cache architecture as a side-channel defence mechanism," *IACR eprint archive*, 2005.

[145] H. Raj, R. Nathuji, A. Singh, and P. England, "Resource management for isolation enhanced cloud services," in *ACM workshop on Cloud computing security*, 2009, pp. 77–84.

[146] M. M. Godfrey and M. Zulkernine, "Preventing cache-based side-channel attacks in a cloud environment," *IEEE transactions on Cloud Computing*, vol. 2, no. 4, pp. 395–408, 2014.

[147] V. Kiriansky, I. Lebedev, S. Amarasinghe, S. Devadas, and J. Emer, "DAWG: A defense against cache timing attacks in speculative execution processors," in *IEEE/ACM Int. Symp. on Microarchitecture (MICRO)*. IEEE, 2018, pp. 974–987.

[148] M. Yan, J. Choi, D. Skarlatos, A. Morrison, C. Fletcher, and J. Torrellas, "InvisiSpec: Making speculative execution invisible in the cache hierarchy," in *IEEE/ACM Int. Symp. on Microarchitecture*. IEEE, 2018, pp. 428–441.

[149] L. Domnitser, A. Jaleel, J. Loew, N. Abu-Ghazaleh, and D. Ponomarev, "Non-monopolizable caches: Low-complexity mitigation of cache side channel attacks," *ACM Transactions on Architecture and Code Optimization*, vol. 8, no. 4, p. 35, 2012.

[150] Intel, CAT, "Improving real-time performance by utilizing cache allocation technology," *Intel Inc.*, 2015.

[151] F. Liu, Q. Ge, Y. Yarom, F. Mckeen, C. Rozas, G. Heiser, and R. B. Lee, "Catalyst: Defeating last-level cache side channel attacks in cloud computing," in *High Performance Computer Architecture, IEEE Int. Symp. on*. IEEE, 2016, pp. 406–418.

[152] A. Ardeshiricham, Y. Takashima, S. Gao, and R. Kastner, "VeriSketch: Synthesizing secure hardware designs with timing-sensitive information flow properties," in *ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1623–1638.

[153] J. P. D. Kaplan, J. Powell, and T. Woller, "AMD memory encryption, white paper," 2016.

[154] B. Gassend, G. E. Suh, D. Clarke, M. Van Dijk, and S. Devadas, "Caches and hash trees for efficient memory integrity verification," in *Int. Symp. on High-Performance Computer Architecture*. IEEE, 2003, pp. 295–306.

[155] O. Goldreich, "Towards a theory of software protection and simulation by oblivious RAMs," in *Annual ACM Symposium on Theory of Computing*, 1987, pp. 182–194.

[156] J. Valamehr, M. Tiwari, T. Sherwood, R. Kastner, T. Huffmire, C. Irvine, and T. Levin, "Hardware assistance for trustworthy systems through 3-D integration," in *Annual Computer Security Applications Conference*, 2010, pp. 199–210.

[157] Altera Inc., "Anti-Tamper Capabilities in FPGA Designs," *Altera White Paper, WP-01066-1.0*, pp. 1–9, 2008, [Online]. Available: https://www.intel.co.uk/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01066-anti-tamper-capabilities-fpga.pdf.

[158] . Semiconductor Components Industries, LLC, "Anti-tamper Active Shield," *ON Semiconductor White Paper TND6321/D Rev. 0*, 2019, [Online]. Available: https://www.onsemi.com/pub/Collateral/TND6321-D.PDF.

[159] R. Prakash, "Anti-tamper Memory," *Cypress Semiconductor Inc. White Paper 001-59690*, 2016, [Online]. Available: https://www.cypress.com/file/99056/download.

[160] N. Burow, S. A. Carr, J. Nash, P. Larsen, M. Franz, S. Brunthaler, and M. Payer, "Control-flow integrity: Precision, security, and performance," *ACM Computing Surveys*, vol. 50, no. 1, pp. 1–33, 2017.

[161] R. de Clercq and I. Verbauwhede, "A survey of hardware-based control flow integrity (CFI)," *arXiv preprint arXiv:1706.07257*, 2017.

[162] D. Cock, Q. Ge, T. Murray, and G. Heiser, "The last mile: An empirical study of timing channels on seL4," in *ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 570–581.

[163] E. Käsper and P. Schwabe, "Faster and timing-attack resistant AES-GCM," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2009, pp. 1–17.

[164] B. Coppens, I. Verbauwhede, K. De Bosschere, and B. De Sutter, "Practical mitigations for timing-based side-channel attacks on modern x86 processors," in *IEEE Symp. on Sec. and Priv.* IEEE, 2009, pp. 45–60.

[165] M. Akkar and C. Giraud, "An implementation of DES and AES, secure against some attacks," in *Cryptographic Hardware and Embedded Systems - CHES, Paris, France*, ser. Lecture Notes in Computer Science, Ç. K. Koç, D. Naccache, and C. Paar, Eds., vol. 2162, no. Generators. Springer, 2001, pp. 309–318.

[166] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, "Higher-order threshold implementations," in *Advances in Cryptology, Int. Conf. on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C.*, ser. Lecture Notes in Computer Science, vol. 8874. Springer, 2014, pp. 326–343.

[167] G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi, and S. Saab, "Test vector leakage assessment (TVLA) methodology in practice," in *Int. Cryptographic Module Conf.*, 2013, p. 13.

[168] T. Zhang, F. Liu, S. Chen, and R. B. Lee, "Side channel vulnerability metrics: The promise and the pitfalls," in *International Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*. New York, NY, USA: ACM, 2013, pp. 2:1–2:8.

[169] H. Kim, S. Hong, B. Preneel, and I. Verbauwhede, "STBC: side channel attack tolerant balanced circuit with reduced propagation delay," in *IEEE Computer Society Annual Symposium on VLSI, ISVLSI, Bochum, Germany*. IEEE, 2017, pp. 74–79.

[170] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *Des. Autom. Test Europe Conf. Exhib. (DATE), Paris, France*. IEEE Computer Society, Feb. 2004, pp. 246–251.

[171] X. Wang, W. Yueh, D. B. Roy, S. Narasimhan, Y. Zheng, S. Mukhopadhyay, D. Mukhopadhyay, and S. Bhunia, "Role of power grid in side channel attack and power-grid-aware secure design," in *Des. Autom. Conf. (DAC), Austin, TX, USA*. ACM, Jun. 2013, pp. 78:1–78:9.

[172] C. Tokunaga and D. T. Blaauw, "Secure AES engine with a local switched-capacitor current equalizer," in *IEEE Int. Solid-State Cir. Conf., San Francisco, CA, USA*. IEEE, 2009, pp. 64–65.

[173] A. Singh, M. Kar, J. H. Ko, and S. Mukhopadhyay, "Exploring power attack protection of resource constrained encryption engines using integrated low-drop-out regulators," in *IEEE/ACM Int. Symp. on Low Power Electronics and Design (ISLPED), Rome, Italy*. IEEE, 2015, pp. 134–139.

[174] M. Kar, A. Singh, S. K. Mathew, A. Rajan, V. De, and S. Mukhopadhyay, "Reducing power side-channel information leakage of AES engines using fully integrated inductive voltage regulator," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 8, pp. 2399–2414, 2018.

[175] D. Canright *et al.*, "A very compact "perfectly masked" S-Box for AES," in *ACNS*, 2008, pp. 446–459.

[176] S. Satpathy, S. Mathew, V. Suresh, and R. Krishnamurthy, "Ultra-low energy security circuits for IoT applications," in *IEEE Int. Conf. on Computer Design (ICCD), Scottsdale, AZ, USA*. IEEE Computer Society, 2016, pp. 682–685.

[177] A. G. Bayrak, F. Regazzoni, D. Novo, and P. Ienne, "Sleuth: Automated verification of software power analysis countermeasures," in *Int. Conf. on Cryptographic Hardware and Embedded Systems (CHES)*. Berlin, Heidelberg: Springer-Verlag, 2013, p. 293–310.

[178] P. Slpsk, P. K. Vairam, C. Rebeiro, and V. Kamakoti, "Karna: A gate-sizing based security aware EDA flow for improved power side-channel attack protection," in *Int. Conf Comput.-Aided Des. (ICCAD)*, Nov. 2019, pp. 1–8.

[179] H. Saputra, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, R. Brooks, S. Kim, and W. Zhang, "Masking the energy behavior of DES encryption [smart cards]," in *Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2003, pp. 84–89.

[180] T. Kim, S. Lee, D. Choi, and H. Yoon, "Protecting secret keys in networked devices with table encoding against power analysis attacks," *Journal of High Speed Networks*, vol. 22, no. 4, pp. 293–307, 2016.

[181] R. Callan, A. Zajic, and M. Prvulovic, "A practical methodology for measuring the side-channel signal available to the attacker for instruction-level events," in *Annual IEEE/ACM Int. Symp. on Microarchitecture*. IEEE, 2014, pp. 242–254.

[182] M. Witteman and M. Oostdijk, "Secure application programming in the presence of side channel attacks," in *RSA conference*, 2008.

[183] C. Kim, M. Schläffer, and S. Moon, "Differential side channel analysis attacks on FPGA implementations of ARIA," *ETRI Journal*, vol. 30, no. 2, pp. 315–325, 2008.

[184] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, Aug. 2002, pp. 13–28.

[185] F.-X. Standaert, T. G. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Annual Int. Conf. on the theory and applications of cryptographic techniques*. Springer, 2009, pp. 443–461.

[186] P. Maistri and R. Leveugle, "Double-data-rate computation as a countermeasure against fault analysis," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1528–1539, 2008.

[187] P. Maistri, P. Vanhauwaert, and R. Leveugle, "A novel double-data-rate AES architecture resistant against fault injection," in *Workshop on Fault Diagnosis and Tolerance in Cryptography*, 2007, pp. 54–61.

[188] R. Karri, G. Kuznetsov, and M. Goessel, "Parity-based concurrent error detection of substitution-permutation network block ciphers," in *Cryptographic Hardware and Embedded Systems - CHES*, C. D. Walter, Ç. K. Koç, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 113–124.

[189] T. G. Malkin, F.-X. Standaert, and M. Yung, "A comparative cost/security analysis of fault attack countermeasures," in *International Workshop on Fault Diagnosis and Tolerance in Cryptography*. Springer, 2006, pp. 159–172.

[190] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, "Security analysis of concurrent error detection against differential fault analysis," *Journal of Cryptographic Engineering*, vol. 5, no. 3, pp. 153–169, 2015.

[191] R. Karri, K. Wu, P. Mishra, and Yongkook Kim, "Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 12, pp. 1509–1517, 2002.

[192] S. Patranabis, A. Chakraborty, D. Mukhopadhyay, and P. P. Chakrabarti, "Fault space transformation: A generic approach to counter differential fault analysis and differential fault intensity analysis on AES-like block ciphers," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 5, pp. 1092–1102, 2017.

[193] M. Agoyan, S. Bouquet, J.-M. Dutertre, J. J.-A. Fournier, J.-B. Rigaud, B. Robisson, and A. Tria, "Design and characterisation of an AES chip embedding countermeasures," *International Journal of Intelligent Engineering Informatics*, p. 00, 2011.

[194] M. Joye, P. Manet, and J. . Rigaud, "Strengthening hardware AES implementations against fault attacks," *IET Information Security*, vol. 1, no. 3, pp. 106–110, 2007.

[195] A. Battistello and C. Giraud, "Fault analysis of infective AES computations," in *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 2013, pp. 101–107.

[196] B. Gierlichs, J.-M. Schmidt, and M. Tunstall, "Infective computation and dummy rounds: Fault protection for block ciphers without check-before-output," in *Progress in Cryptology – LATINCRYPT 2012*, A. Hevia and G. Neven, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 305–321.

[197] J. Breier and W. He, "Multiple fault attack on PRESENT with a hardware Trojan implementation in FPGA," in *2015 International Workshop on Secure Internet of Things (SIoT)*, 2015, pp. 58–64.

[198] I. Roy, C. Rebeiro, A. Hazra, and S. Bhunia, "SAFARI: Automatic synthesis of fault-attack resistant block cipher implementations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 4, pp. 752–765, 2020.

[199] S. Saha, D. Mukhopadhyay, and P. Dasgupta, "ExpFault: An automated framework for exploitable fault characterization in block ciphers," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 2, pp. 242–276, 2018.

[200] M. Srivatsava, P. Slpsk, C. Rebeiro, A. Hazra, and S. Bhunia, "Solomon: An automated framework for detecting fault attack vulnerabilities in hardware," in *Des. Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2020.

[201] K. K, I. Roy, C. Rebeiro, A. Hazra, and S. Bhunia, "FEDS: Comprehensive fault attack exploitability detection for software implementations of block ciphers," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 2, pp. 272–299, 2020.

[202] F. Koushanfar, I. Hong, and M. Potkonjak, "Behavioral synthesis techniques for intellectual property protection," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 10, no. 3, p. 523–545, 2005.

[203] D. Kirovski, Y. Hwang, M. Potkonjak, and J. Cong, "Protecting combinational logic synthesis solutions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2687–2696, 2006.

[204] A. Cui, C. H. Chang, and S. Tahar, "IP watermarking using incremental technology mapping at logic synthesis level," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 9, pp. 1565–1570, 2008.

[205] A. Sengupta and M. Rathor, "Crypto-based dual-phase hardware steganography for securing IP cores," *IEEE Letters of the Computer Society*, vol. 2, no. 4, pp. 32–35, 2019.

[206] A. Sengupta and M. Rathor, "Structural obfuscation and crypto-steganography-based secured JPEG compression hardware for medical imaging systems," *IEEE Access*, vol. 8, pp. 6543–6565, 2020.

[207] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Des. Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2008, pp. 1069–1074.

[208] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic encryption: A fault analysis perspective," in *Des. Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2012, pp. 953–958.

[209] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," *IEEE Des. Test. Comput.*, vol. 27, no. 1, pp. 66–75, 2010.

[210] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *IEEE Int. Symp. on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 137–143.

[211] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 517–532, 2020.

[212] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel bypass attack and BDD-based tradeoff analysis against all known logic locking attacks," in *Cryptographic Hardware and Embedded Systems – CHES 2017*, W. Fischer and N. Homma, Eds. Cham: Springer International Publishing, 2017, pp. 189–210.

[213] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *ACM Conf. on Comp. and Comm. Sec. (CCS)*. New York, NY, USA: ACM, 2017, p. 1601–1618.

[214] F. Yang, M. Tang, and O. Sinanoglu, "Stripped functionality logic locking with hamming distance-based restore unit (SFLL-hd) – unlocked," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 10, pp. 2778–2786, 2019.

[215] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2514–2527, 2020.

[216] H. M. Kamali, K. Z. Azar, H. Homayoun, and A. Sasan, "Full-lock: Hard distributions of sat instances for obfuscating circuits using fully configurable logic and routing blocks," in *Des. Autom. Conf. (DAC)*. New York, NY, USA: ACM, Jun. 2019.

[217] R. S. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1493–1502, 2009.

[218] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: Identification of stealthy malicious logic using boolean functional analysis," in *ACM Conf. on Comp. and Comm. Sec. (CCS)*. New York, NY, USA: ACM, 2013, p. 697–708.

[219] J. Zhang, F. Yuan, L. Wei, Y. Liu, and Q. Xu, "VeriTrust: Verification for hardware trust," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 7, pp. 1148–1161, 2015.

[220] S. K. Haider, C. Jin, M. Ahmad, D. M. Shila, O. Khan, and M. van Dijk, "Advancing the state-of-the-art in hardware Trojans detection,"

IEEE Transactions on Dependable Secure Computing, vol. 16, no. 1, pp. 18–32, 2019.

[221] M. Oya, Y. Shi, M. Yanagisawa, and N. Togawa, "A score-based classification method for identifying hardware-Trojans at gate-level netlists," in *Des. Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2015, pp. 465–470.

[222] M. Rathmair, F. Schupfer, and C. Krieg, "Applied formal methods for hardware Trojan detection," in *IEEE Symp. On Cir. and Syst. (ISCAS)*, 2014, pp. 169–172.

[223] X. Guo, R. G. Dutta, P. Mishra, and Y. Jin, "Scalable SoC trust verification using integrated theorem proving and model checking," in *IEEE Int. Symp. on Hardware Oriented Security and Trust (HOST)*, 2016, pp. 124–129.

[224] W. Hu, B. Mao, J. Oberg, and R. Kastner, "Detecting hardware Trojans with gate-level information-flow tracking," *Computer*, vol. 49, no. 8, pp. 44–52, 2016.

[225] Y. Huang, S. Bhunia, and P. Mishra, "MERS: Statistical test generation for side-channel analysis based Trojan detection," in *ACM Conf. on Comp. and Comm. Sec. (CCS)*. New York, NY, USA: ACM, 2016, p. 130–141.

[226] M. Banga, M. Chandrasekar, L. Fang, and M. S. Hsiao, "Guided test generation for isolation and detection of embedded Trojans in ICs," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*. New York, NY, USA: ACM, 2008, p. 363–366.

[227] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *IEEE Symp. on Sec. and Priv. (SP)*, 2007, pp. 296–310.

[228] K. Xiao, X. Zhang, and M. Tehranipoor, "A clock sweeping technique for detecting hardware Trojans impacting circuits delay," *IEEE Design & Test*, vol. 30, no. 2, pp. 26–34, 2013.

[229] B. Zhou, R. Adato, M. Zangeneh, T. Yang, A. Uyar, B. Goldberg, S. Unlu, and A. Joshi, "Detecting hardware Trojans using backside optical imaging of embedded watermarks," in *Des. Autom. Conf. (DAC)*. New York, NY, USA: ACM, Jun. 2015.

[230] D. Forte, C. Bao, and A. Srivastava, "Temperature tracking: An innovative run-time approach for hardware Trojan detection," in *Int. Conf Comput.-Aided Des.* IEEE, 2013, pp. 532–539.

[231] S. Narasimhan, D. Du, R. S. Chakraborty, S. Paul, F. G. Wolff, C. A. Papachristou, K. Roy, and S. Bhunia, "Hardware Trojan detection by multiple-parameter side-channel analysis," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2183–2195, 2013.

[232] Y. Huang, S. Bhunia, and P. Mishra, "Scalable test generation for Trojan detection using side channel analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 11, pp. 2746–2760, 2018.

[233] Y. Liu, K. Huang, and Y. Makris, "Hardware Trojan detection through golden chip-free statistical side-channel fingerprinting," in *Des. Autom. Conf. (DAC)*. New York, NY, USA: ACM, Jun. 2014, p. 1–6.

[234] T. Hoque, S. Narasimhan, X. Wang, S. Mal-Sarkar, and S. Bhunia, "Golden-Free hardware Trojan detection with high sensitivity under process noise," *J. Electron. Test.*, vol. 33, no. 1, p. 107–124, 2017.

[235] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A novel technique for improving hardware Trojan detection and reducing Trojan activation time," *IEEE Trans. VLSI Syst.*, vol. 20, no. 1, pp. 112–125, 2012.

[236] J. Rajendran, V. Jyothi, O. Sinanoglu, and R. Karri, "Design and analysis of ring oscillator based Design-for-Trust technique," in *VLSI Test Symposium*, 2011, pp. 105–110.

[237] Y. Cao, C. H. Chang, and S. Chen, "A cluster-based distributed active current sensing circuit for hardware Trojan detection," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 12, pp. 2220–2231, 2014.

[238] K. Xiao, D. Forte, and M. Tehranipoor, "A novel built-in self-authentication technique to prevent inserting hardware Trojans," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 12, pp. 1778–1791, 2014.

[239] S. Bhunia, M. Abramovici, D. Agrawal, P. Bradley, M. S. Hsiao, J. Plusquellic, and M. Tehranipoor, "Protection against hardware Trojan attacks: Towards a comprehensive solution," *IEEE Design & Test*, vol. 30, no. 3, pp. 6–17, 2013.

[240] Y. Jin and D. Sullivan, "Real-time trust evaluation in integrated circuits," in *Des. Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2014, pp. 1–6.

[241] J. He, X. Guo, H. Ma, Y. Liu, Y. Zhao, and Y. Jin, "Runtime trust evaluation and hardware Trojan detection using on-chip EM sensors," in *Des. Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[242] J. Rajendran, V. Vedula, and R. Karri, "Detecting malicious modifications of data in third-party intellectual property cores," in *Des. Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.

[243] N. Veeranna and B. C. Schafer, "Hardware Trojan detection in behavioral intellectual properties (IP's) using property checking techniques," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 4, pp. 576–585, 2017.

[244] Y. Jin and Y. Makris, "Proof carrying-based information flow tracking for data secrecy protection and hardware trust," in *IEEE VLSI Test Symposium (VTS)*, 2012, pp. 252–257.

[245] M. Bidmeshki and Y. Makris, "Toward automatic proof generation for information flow policies in third-party hardware IP," in *IEEE Int. Symp. on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 163–168.

[246] Y. Jin, X. Guo, R. G. Dutta, M. Bidmeshki, and Y. Makris, "Data secrecy protection through information flow tracking in proof-carrying hardware IP—part I: Framework fundamentals," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2416–2429, 2017.

[247] M. Bidmeshki, X. Guo, R. G. Dutta, Y. Jin, and Y. Makris, "Data secrecy protection through information flow tracking in proof-carrying hardware IP—part II: Framework automation," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2430–2443, 2017.

[248] J. Portillo, E. John, and S. Narasimhan, "Building trust in 3PIP using asset-based security property verification," in *IEEE VLSI Test Symposium (VTS)*, 2016, pp. 1–6.

[249] X. Chen, Q. Liu, S. Yao, J. Wang, Q. Xu, Y. Wang, Y. Liu, and H. Yang, "Hardware Trojan detection in third-party digital intellectual property cores by multilevel feature analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 7, pp. 1370–1383, 2018.

[250] L. Piccolboni, A. Menon, and G. Pravadelli, "Efficient control-flow subgraph matching for detecting hardware Trojans in RTL models," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 5s, 2017.

[251] A. Malekpour, R. Ragel, A. Ignjatovic, and S. Parameswaran, "Dosguard: Protecting pipelined mpsocs against hardware Trojan based DoS attacks," in *IEEE Int. Conf. on Application-specific Systems, Architectures and Processors (ASAP)*, 2017, pp. 45–52.

[252] S. Dupuis, P. Ba, G. Di Natale, M. Flottes, and B. Rouzeyre, "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware Trojans," in *IEEE International On-Line Testing Symposium (IOLTS)*, 2014, pp. 49–54.

[253] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 410–424, 2015.

[254] Y. Xie, C. Bao, and A. Srivastava, "Security-aware design flow for 2.5D IC technology," in *International Workshop on Trustworthy Embedded Devices*. New York, NY, USA: ACM, 2015, p. 31–38.

[255] Y. Lao and K. K. Parhi, "Obfuscating DSP circuits via high-level transformations," *IEEE Trans. VLSI Syst.*, vol. 23, no. 5, pp. 819–830, 2015.

[256] A. Sengupta, D. Roy, S. P. Mohanty, and P. Corcoran, "DSP design protection in CE through algorithmic transformation based structural obfuscation," *IEEE Trans. Consum. Electron.*, vol. 63, no. 4, pp. 467–476, 2017.

[257] A. Sengupta, D. Roy, S. P. Mohanty, and P. Corcoran, "Low-cost obfuscated JPEG CODEC IP core for secure CE hardware," *IEEE Trans. Consum. Electron.*, vol. 64, no. 3, pp. 365–374, 2018.

[258] A. Sengupta, S. Neema, P. Sarkar, S. Harsha P, S. P. Mohanty, and M. K. Naskar, "Obfuscation of fault secured DSP design through hybrid transformation," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jul. 2018, pp. 732–737.

[259] K. Huang, J. M. Carulli, and Y. Makris, "Parametric counterfeit IC detection via support vector machines," in *Int. Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, Oct. 2012, pp. 7–12.

[260] W. Shan, S. Zhang, J. Xu, M. Lu, L. Shi, and J. Yang, "Machine learning assisted side-channel-attack countermeasure and its application on a 28-nm AES circuit," *IEEE J. Solid-State Circuits*, 2019.

[261] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE Symp. on Sec. and Priv. (SP)*, May 2016, pp. 582–597.

[262] A. N. Bhagoji, D. Cullina, C. Sitawarin, and P. Mittal, "Enhancing robustness of machine learning systems via data transformations," in *Annual Conference on Information Sciences and Systems*, Nov. 2018, pp. 1–5.

[263] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *ACM Conf. on Comp. and Comm. Sec. (CCS)*. New York, NY, USA: ACM, Oct. 2017, p. 135–147.

[264] J. Lin, C. Gan, and S. Han, "Defensive quantization: When efficiency meets robustness," Int. Conf. on Learning Representations (ICLR), New Orleans, LA, USA, 2019.

[265] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," 2017.

[266] K. W. Nixon, J. Mao, J. Shen, H. Yang, H. H. Li, and Y. Chen, "SPN dash: Fast detection of adversarial attacks on mobile via sensor pattern noise fingerprinting," in *Int. Conf Comput.-Aided Des. (ICCAD)*. New York, NY, USA: ACM, Nov. 2018.

[267] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," 2017.

[268] J. Wang, G. Dong, J. Sun, X. Wang, and P. Zhang, "Adversarial sample detection for deep neural network through model mutation testing," in *Int. Conf. on Software Engineering (ICSE*. IEEE, Aug. 2019, p. 1245–1256.

[269] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Network and Distributed Systems Security Symposium (NDSS-2018)*, Jan. 2018.

[270] S. Wang, W. Liu, and C. H. Chang, "Detecting adversarial examples for deep neural networks via layer directed discriminative noise injection," in *Asian Hardware Oriented Security and Trust Symposium (AsianHOST-2019)*, Xi'an, China, Dec. 2019, pp. 1–6.

[271] ——, "Fired neuron rate based decision tree for detection of adversarial examples in DNNs," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, Oct. 2020.

[272] B. D. Rouhani, M. Samragh, M. Javaheripi, T. Javidi, and F. Koushanfar, "DeepFense: Online accelerated defense against adversarial deep learning," in *Int. Conf Comput.-Aided Des. (ICCAD)*, San Diego, California, Nov. 2018.

[273] X. Wang, R. Hou, B. Zhao, F. Yuan, J. Zhang, D. Meng, and X. Qian, "DNNGuard: An elastic heterogeneous DNN accelerator architecture against adversarial attacks," in *Int. Conf. on Arch. Support for Prog. Lang. and Oper. Sys. (ASPLOS)*, Lausanne, Switzerland, Mar. 2020, pp. 19—-34.

[274] M. Tiwari, H. M. Wassel, B. Mazloom, S. Mysore, F. T. Chong, and T. Sherwood, "Complete information flow tracking from the gates up," in *Int. Conf. on Arch. Support for Prog. Lang. and Oper. Sys. (ASPLOS)*, 2009, pp. 109–120.

[275] W. Hu, J. Oberg, A. Irturk, M. Tiwari, T. Sherwood, D. Mu, and R. Kastner, "Theoretical fundamentals of gate level information flow tracking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 8, pp. 1128–1140, 2011.

[276] W. Hu, D. Mu, J. Oberg, B. Mao, M. Tiwari, T. Sherwood, and R. Kastner, "Gate-level information flow tracking for security lattices," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 20, no. 1, pp. 1–25, 2014.

[277] W. Hu, J. Oberg, A. Irturk, M. Tiwari, T. Sherwood, D. Mu, and R. Kastner, "On the complexity of generating gate level information flow tracking logic," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 1067–1080, 2012.

[278] M. Tiwari, J. K. Oberg, X. Li, J. Valamehr, T. Levin, B. Hardekopf, R. Kastner, F. T. Chong, and T. Sherwood, "Crafting a usable microkernel, processor, and I/O system with strict and provable information flow security," in *Annual Int. Symp. on Computer Architecture (ISCA)*, 2011, pp. 189–199.

[279] A. Ardeshiricham, W. Hu, J. Marxen, and R. Kastner, "Register transfer level information flow tracking for provably secure hardware design," in *Des. Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2017, pp. 1691–1696.

[280] A. Ardeshiricham, W. Hu, and R. Kastner, "Clepsydra: Modeling timing flows in hardware designs," in *Int. Conf Comput.-Aided Des. (ICCAD)*, Nov. 2017, pp. 147–154.

[281] X. Guo, H. Zhu, Y. Jin, and X. Zhang, "When capacitors attack: Formal method driven design and detection of charge-domain Trojans," in *Des. Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2019, pp. 1727–1732.

[282] D. Zhang, Y. Wang, G. E. Suh, and A. C. Myers, "A hardware design language for timing-sensitive information-flow security," in *Int. Conf. on Arch. Support for Prog. Lang. and Oper. Sys. (ASPLOS)*. New York, NY, USA: ACM, 2015, pp. 503–516.

[283] J. A. Goguen and J. Meseguer, "Security policies and security models," in *IEEE Symposium on Security & Privacy*, 1982, pp. 11–20.

[284] A. Ferraiuolo, Weizhe Hua, A. C. Myers, and G. E. Suh, "Secure information flow verification with mutable dependent types," in *Des. Autom. Conf. (DAC)*, Jun. 2017, pp. 1–6.

[285] S. Deng, D. Gümüşoğlu, W. Xiong, S. Sari, Y. S. Gener, C. Lu, O. Demir, and J. Szefer, "Secchisel framework for security verification of secure processor architectures," in *International Workshop on Hardware and Architectural Support for Security and Privacy (HASPP)*. New York, NY, USA: ACM, 2019, pp. 7:1–7:8.

[286] Mentor Graphics, "Questa secure check - exhaustive verification of secure paths to critical hardware storage," 2016, https://www.mentor.com/products/fv/questa-secure-check.

[287] Cadence, "JasperGold security path verification App," 2016, https://www.cadence.com/content/cadence-www/global/en_US/home/tools/system-design-and-verification/formal-and-static-verification/jasper-gold-verification-platform/security-path-verification-app.html.

[288] G. Cabodi, P. Camurati, S. F. Finocchiaro, C. Loiacono, F. Savarese, and D. Vendraminetto, "Secure path verification," in *IEEE International Verification and Security Workshop (IVSW)*, 2016, pp. 1–6.

[289] Tortuga Logic, "Radix-S hardware root of trust security verification framework," 2019, https://www.tortugalogic.com/radix-s/.

[290] Tortuga Logic, "Radix-M hardware security platform for firmware security validation," 2019, https://www.tortugalogic.com/radix-m/.

[291] G. Dessouky, D. Gens, P. Haney, G. Persyn, A. Kanuparthi, H. Khattri, J. M. Fung, A.-R. Sadeghi, and J. Rajendran, "Hardfails: Insights into software-exploitable hardware bugs," in *USENIX Conference on Security Symposium*. USA: USENIX Association, 2019, p. 213–230.

[292] Synopsys, "CustomSim Reliability Analysis," 2019, https://www.synopsys.com/verification/ams-verification/reliability-analysis/customsim-reliability-analysis.html.

[293] ——, "VC Functional Safety Manager," 2019, https://www.synopsys.com/verification/vc-functional-safety-manager.html.

[294] J. Urdahl, S. Udupi, T. Ludwig, D. Stoffel, and W. Kunz, "Properties first? A new design methodology for hardware, and its perspectives in safety analysis," in *Int. Conf Comput.-Aided Des. (ICCAD)*. New York, NY, USA: ACM, Nov. 2016.

[295] W. Hu, A. Ardeshiricham, M. S. Gobulukoglu, X. Wang, and R. Kastner, "Property specific information flow analysis for hardware security verification," in *Int. Conf Comput.-Aided Des. (ICCAD)*, Nov. 2018, pp. 1–8.

[296] J. He, H. Ma, X. Guo, Y. Zhao, and Y. Jin, "Design for em side-channel security through quantitative assessment of rtl implementations," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 62–67.

[297] S. Takarabt, K. Chibani, A. Facon, S. Guilley, Y. Mathieu, L. Sauvage, and Y. Souissi, "Pre-silicon embedded system evaluation as new eda tool for security verification," in *IEEE International Verification and Security Workshop (IVSW)*, 2018, pp. 74–79.

[298] Y. Hu, V. V. Menon, A. Schmidt, J. Monson, M. French, and P. Nuzzo, "Security-driven metrics and models for efficient evaluation of logic encryption schemes," in *ACM/IEEE Int. Conf. on Formal Methods and Models for System Design (MEMOCODE)*. New York, NY, USA: ACM, 2019.

[299] S. Patnaik, M. Ashraf, O. Sinanoglu, and J. Knechtel, "Best of both worlds: Integration of split manufacturing and camouflaging into a security-driven CAD flow for 3D ICs," in *Int. Conf Comput.-Aided Des.(ICCAD)*. New York, NY, USA: ACM, Nov. 2018.

[300] R. Kastner, W. Hu, and A. Althoff, "Quantifying hardware security using joint information flow analysis," in *Des. Autom. Test Europe Conf. Exhib. (DATE)*. IEEE, Mar. 2016, pp. 1523–1528.

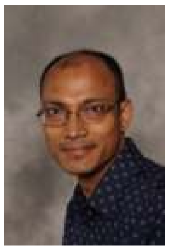
Wei Hu (M'17) is currently an associate professor with the School of Cybersecurity, Northwestern Polytechnical University (NPU). He got his BS, MS and PhD in 2005, 2008 and 2012 respectively all from the same university. His research interests are in hardware security, cryptography, formal security verification, logic and high-level synthesis, formal methods and reconfigurable computing.

Dr. Hu serves as the Guest Associate Editor of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. He has been an Organizing Committee member of IEEE International Symposium on Hardware Oriented Security and Trust and Asian Hardware Oriented Security and Trust Symposium since 2017. He was the Technical Program Co-Chair of 2019 Asian Hardware Oriented Security and Trust Symposium and Technical Program Committee Member of ICCD, ASAP and CFTC. He published over 70 papers in peer-reviewed journals and conferences, 2 books and 3 patents.

Chip Hong Chang (S'92-M'98-SM'03-F'18) received the B. Eng. (Hons.) degree from the National University of Singapore in 1989, and the M. Eng. and Ph.D. degrees from Nanyang Technological University (NTU) of Singapore in 1993 and 1998, respectively. He is an Associate Professor of the School of Electrical and Electronic Engineering (EEE) of NTU. He held joint appointments with the university as Assistant Chair of Alumni from 2008 to 2014, Deputy Director of the Center for High Performance Embedded Systems from 2000 to 2011, and Program Director of the Center for Integrated Circuits and Systems from 2003 to 2009. He has co-edited 5 books, published 13 book chapters, more than 100 international journal papers (more than 70 are in IEEE) and more than 180 refereed international conference papers (mostly in IEEE), and delivered over 40 colloquia. His current research interests include hardware security, DNN security, unconventional number systems, low-power arithmetic circuits, digital image and signal processing algorithms and architectures.

Dr. Chang serves as the Senior Associate Editor and Associate Editor of the IEEE Transactions on Information Forensics and Security since June 2020, and 2016-2019, respectively, Associate Editor of IEEE Transactions on Very Large Scale Integration (VLSI) Systems from 2011 to 2020, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems from 2016 to 2019, IEEE Transactions on Circuits and Systems-I since 2020 and from 2010 to 2013, IEEE Access from 2013 to 2019, Integration, the VLSI Journal from 2013-2015, Springer Journal of Hardware and System Security from 2016 to 2020 and Microelectronics Journal from 2014 to 2020. He guest edited several journal special issues and served in the organizing and technical program committee of more than 60 international conferences. He is also an IET Fellow and 2018-2019 Distinguished Lecturer of IEEE Circuits and Systems Society.

Anirban Sengupta is an Associate Professor in Computer Science and Engineering at Indian Institute of Technology (I.I.T) Indore. He is an elected Fellow of IET and Fellow of British Computer Society (FBCS), UK. He is a registered Professional Engineer of Ontario (P.Eng.). He has been awarded prestigious IEEE Distinguished Lecturer by IEEE Consumer Electronics Society in 2017 and IEEE Distinguished Visitor by IEEE Computer Society in 2019. He has more than 216 Publications including 3 Books and 11 Patents. He is author of 3 Books from IET and Springer on Hardware Security, IP core protection and VLSI Design. He is currently Deputy Editor-in-Chief of IET Computers and Digital Techniques Journal and Editor-in-Chief of IEEE VLSI Circuits & Systems Letter of IEEE Computer Society TCVLSI. He is currently the Chair of IEEE Computer Society TCVLSI. He currently serves/served in more than 16 Editorial positions of several IEEE Transactions/Journals, IET and Elsevier Journals including IEEE Transactions on Aerospace and Electronic Systems (TAES), IEEE Transactions on VLSI Systems, IEEE Transactions on Consumer Electronics, IEEE Access Journal, IET Journal on Computer & Digital Techniques, IEEE Consumer Electronics, IEEE Canadian Journal of Electrical and Computer Engineering, IEEE VLSI Circuits & Systems Letter and other Journals. He is recipient of several IEEE Honors such as IEEE Chester Sall Memorial Consumer Electronics Award, IEEE Outstanding Editor Awards, IEEE Outstanding Service Awards and IEEE Best Paper Awards from Journals/Magazines and Conferences. He was the General/Conference Chair of 37th IEEE International Symposium on Consumer Electronics (ICCE) 2019, Las Vegas and Technical Program Chair of 36th IEEE International Conference on Consumer Electronics (ICCE) 2018 in Las Vegas, 9th IEEE International Conference on Consumer Electronics (ICCE) - Berlin 2019, 15th IEEE International Conference on Information Technology (ICIT) 2016, 3rd IEEE International Symposium on Nanoelectronic and Information Systems (iNIS) 2017. More details are available at: www.anirban-sengupta.com



Swarup Bhunia received his B.E. (Hons.) from Jadavpur University, Kolkata, India, M.Tech. from the Indian Institute of Technology (IIT), Kharagpur, and Ph.D. from Purdue University, IN, USA. Currently, Dr. Bhunia is a professor and Semmoto Endowed Chair in the University of Florida, FL, USA. Earlier he was appointed as the T. and A. Schroeder associate professor of Electrical Engineering and Computer Science at Case Western Reserve University, Cleveland, OH, USA. He has over ten years of research and development experience with over 250 publications in peer-reviewed journals and premier conferences. His research interests include hardware security and trust, adaptive nanocomputing and novel test methodologies. Dr. Bhunia received IBM Faculty Award (2013), National Science Foundation career development award (2011), Semiconductor Research Corporation Inventor Recognition Award (2009), and SRC technical excellence award as a team member (2005), and several best paper awards/nominations. He has been serving as an associate editor of IEEE Transactions on CAD, IEEE Transactions on Multi-Scale Computing Systems, ACM Journal of Emerging Technologies; served as guest editor of IEEE Design & Test of Computers (2010, 2013) and IEEE Journal on Emerging and Selected Topics in Circuits and Systems (2014). He has served in the organizing and program committee of many IEEE/ACM conferences.

Ryan Kastner is a professor in the Department of Computer Science and Engineering at the University of California San Diego. He received a PhD in Computer Science (2002) at UCLA, a Masters degree in engineering (2000) and Bachelor degrees (BS) in both Electrical Engineering and Computer Engineering (1999) from Northwestern University. Professor Kastner's research interests are broad and varied, but generally fall into three areas: hardware acceleration, hardware security, and remote sensing. He is the co-director of the Wireless Embedded Systems Graduate Program — a specialized Masters degree targeting individuals working in local industries. He co-directs the Engineers for Exploration Program, which pairs student researchers with domain scientists to build technologies to aid in activities related to archaeology, conservation, and cultural heritage. He has been working the hardware security space for over 15 years performing fundamental research in FPGA security, 3D integrated circuit security, and hardware information flow tracking. He is the co-founder of the company Tortuga Logic that develops hardware security solutions based upon technology developed in his research group.

Hai (Helen) Li (M'08-SM'16-F'19) received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, and the Ph.D. degree from the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, in 2004. Dr. Li currently is a Professor of the Department of Electrical and Computer Engineering at Duke University, Durham, NC, USA. She has authored or co-authored more than 200 papers in peer-reviewed journals and conferences and a book entitled Nonvolatile Memory Design: Magnetic, Resistive, and Phase Changing (CRC Press, 2011). Her current research interests include neuromorphic computing systems, machine learning and deep neural networks, memory design and architecture, and cross-layer optimization for low power and high performance. Dr. Li is a Distinguished Lecturer of the IEEE CAS society (2018-2019) and a distinguished speaker of ACM (2017-2020). Dr. Li is a recipient of the NSF Career Award (2012), DARPA Young Faculty Award (2013), TUM-IAS Hans Fischer Fellowship from Germany (2017), and ELATE Fellowship (2020). She received eight best paper awards and additional nine best paper nominations from international conferences. Dr. Li serves as Associate Editor of IEEE TCAD, IEEE TVLSI, IEEE TCAS-II, IEEE TMSCS, ACM TECS, IEEE CEM, ACM TODAES, and IET-CPS. She was the General Chair or Technical Program Chair of multiple IEEE/ACM conferences and the Technical Program Committee members of over 30 international conference series. Dr. Li is an IEEE fellow and a distinguished member of the ACM.