CHAPTER

INTRODUCTION TO HARDWARE SECURITY

1

CONTENTS

Computer security has become an essential part of the modern electronic world. Hardware security, which deals with the security of electronic hardware, encompassing its architecture, implementation, and validation, has evolved alongside it into an important field of computer security. In the context of this book, "hardware" indicates electronic hardware. Like any field of security, the topic of hardware security focuses on attacks crafted to steal or compromise assets and approaches designed to protect these assets. The assets under consideration are the hardware components themselves, for instance, integrated circuits (ICs) of all types, passive components (such as, resistors, capacitors, inductors), and printed circuit boards (PCBs); as well as the secrets stored inside these components, for instance, cryptographic keys, digital rights management (DRM) keys, programmable fuses, sensitive user data, firmware, and configuration data.

Figure 1.1 illustrates different fields of security related to a modern computing system. Network security focuses on the attacks on a network connecting multiple computer systems, and the mechanisms

Hardware Security. https://doi.org/10.1016/B978-0-12-812477-2.00006-X

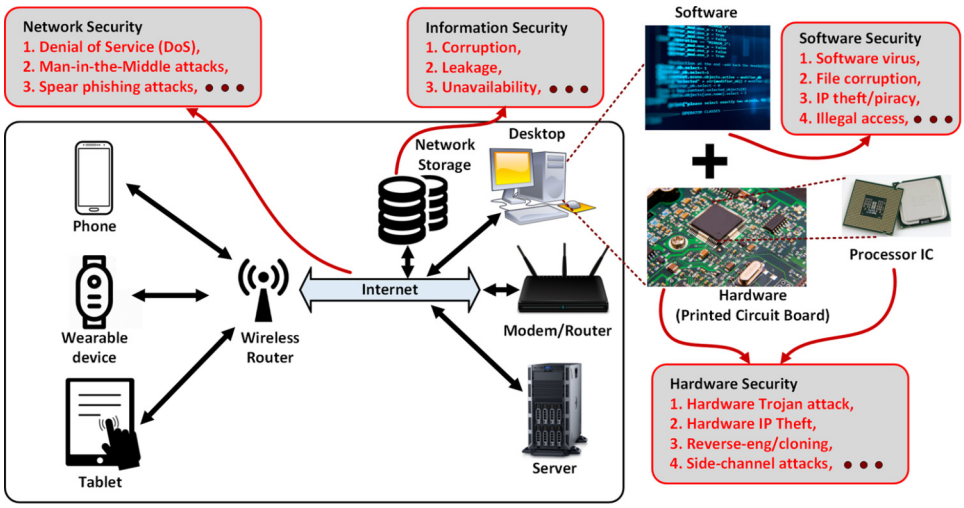

FIGURE 1.1

The landscape of security in modern computing systems.

to ensure its usability and integrity under potential attacks. Software security focuses on malicious attacks on software, often exploiting different implementation bugs, such as inconsistent error handling and buffer overflows, and techniques to ensure reliable software operation in presence of potential security risks. Information security focuses on the general practice of providing confidentiality, integrity, and availability of information through protection against unauthorized access, use, modification, or destruction. Hardware security, on the other hand, focuses on attacks and protection of hardware. It forms the foundation of system security, providing trust anchor for other components of a system that closely interact with it. The remaining chapters of the book illustrate how a variety of attacks on hardware challenge this notion, and how effective countermeasures against these attacks can be employed to ensure the security and trust of hardware.

The book covers all topics related to electronic hardware and systems security encompassing various application domains, including embedded systems, cyber-physical systems (CPS), internet of things (IoT), and biomedical systems (for example, implants and wearables). It describes security and trust issues, threats, attacks, vulnerabilities, protection approaches, including design, validation, and trust monitoring solutions for hardware at all levels of abstraction: from hardware intellectual properties (IPs) to ICs to PCBs and systems. The coverage also includes associated metrics, tools, and benchmarks.

1.1 OVERVIEW OF A COMPUTING SYSTEM

A computing system is a system of interconnected components. The following highlights the major components in such a system and their roles: memory for information storage; processor for information processing, and input/output devices (for example, peripheral devices, such as keyboards, printers, and displays) for interfacing with human users or other systems. These systems are capable of capturing

and transforming information; and communicating them with other computing systems. Information storage and processing are often performed on digital data. However, in many applications, there is an analog front-end that acquires analog signals from the physical world, conditions and then digitizes them. A digital processing unit then performs specific operations on the digital form. Optionally, a back-end unit then transforms the processed digital signal into analog to interface with the physical world. Traditionally, computing systems have been broadly classified into two categories: (a) general-purpose systems and (b) embedded systems. The first category included systems, such as desktop, laptop, and servers, which had the following characteristics: (1) complex and optimized architecture, (2) versatile and easily programmable, and (3) suitable for diverse use-case scenarios. On the other hand, the second category included systems, such as digital cameras, home automation devices, wearable health monitors, and biomedical implants, which have the following characteristics: (1) highly customized design, (2) tight hardware-software integration, and (3) unique use-case constraints.

Over the years, the gap between these two categories narrowed with embedded systems becoming more flexible, and having more computing power to handle general-purpose applications. Two new classes of systems have emerged, which borrow features from both categories: (1) cyber-physical systems and (2) internet of things. In the first class, computer-based information processing systems are deeply intertwined with the Internet and its users, and the physical world. Examples of such systems include smart grid, autonomous vehicles, and robotic systems. The second class, on the other hand, includes computing systems that connect to the Internet, the cloud, and other endpoint devices, and interact with the physical world by collecting and exchanging data using embedded sensors and controlling physical devices through actuators. Such devices include smart home automation devices and personal health monitors. Both classes of devices increasingly rely on artificial intelligence to make autonomous decisions, to have situational awareness, and to better respond to different usage patterns through learning. The distinction between these two classes is getting blurred gradually, with CPS having similar characteristics as IoT devices. Devices falling into these classes share many features, which have security implications, such as, (1) long and complex life, during which the security requirements may change; (2) machine-to-machine communication without any human in the loop, which may create an insecure communication link, and need for novel authentication approaches; and (3) mass production in the millions with identical configuration, which can help an attacker identify vulnerabilities of one device, and use that knowledge to break into many.

Moreover, modern computing systems usually do not operate in isolation. They are connected with other computers and/or the cloud, which is a collection of computers that provides shared computing or storage resources to a bunch of other computers. Figure 1.2 shows different components of a modern

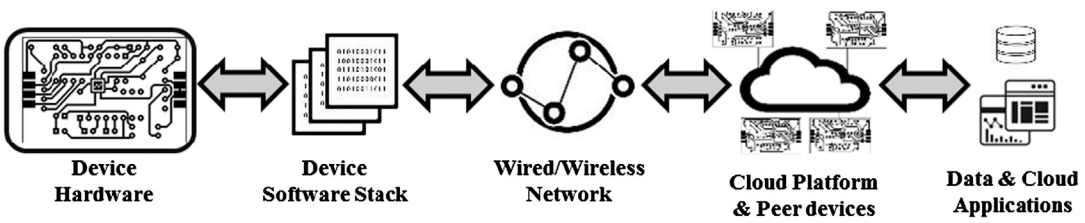

FIGURE 1.2

Different layers in the organization of modern computing systems.

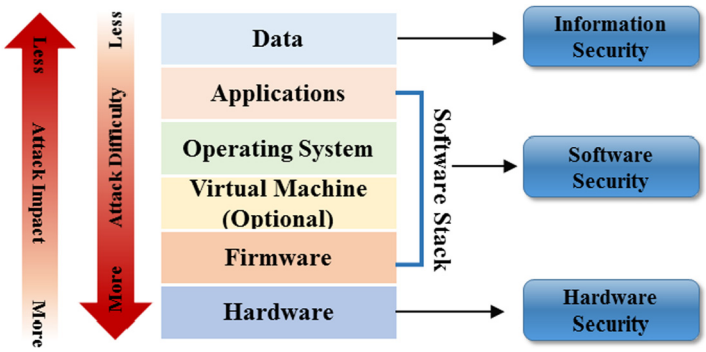

FIGURE 1.3

Attack impact and difficulty at different layers of a computing system.

computing system, for example, a CPS or IoT system, starting from hardware units to cloud and the data/applications in the cloud. Each component in this organization is associated with diverse security issues and corresponding solutions. The weakest link in this complex, often physically distributed system usually determines the security of the whole system. Achieving security of the entire system requires a significant rethinking on how to integrate specific security solutions for each component into a holistic protection approach.

1.2 LAYERS OF A COMPUTING SYSTEM

Modern computing systems can be viewed as an organization consisting of multiple layers of abstraction, as illustrated in Fig. 1.3. The hardware layer lies at the bottom of it, followed by the firmware that interfaces with the physical hardware layer. The firmware layer is followed by the software stack, comprising of an optional virtualization layer, the operating system (OS), and then the application layer. All types of computing systems discussed in the previous sections share this common structure. The data being processed by a computing system is stored in the hardware layer in volatile (for example, static or dynamic random access memory) or non-volatile (such as NAND or NOR flash) memory and accessed by the software layers. A system is connected to another system or to the Internet using networking mechanisms that are realized by a combination of hardware and software components. Computer security issues span all these layers. While hardware security issues are relatively fewer than those at other layers (as shown in Fig. 1.3), they usually have much larger impacts on system security. In particular, they typically affect a much larger number of devices than security issues in software and network, as manifested by the recent discoveries, such as the Spectre and Meltdown bugs [9] in modern processors.

1.2.1 ELECTRONIC HARDWARE

The hardware in a computing system can, itself, be viewed as consisting of three layers, as illustrated in Fig. 1.4. At the top of it, we have a system-level hardware, that is, the integration of all physical

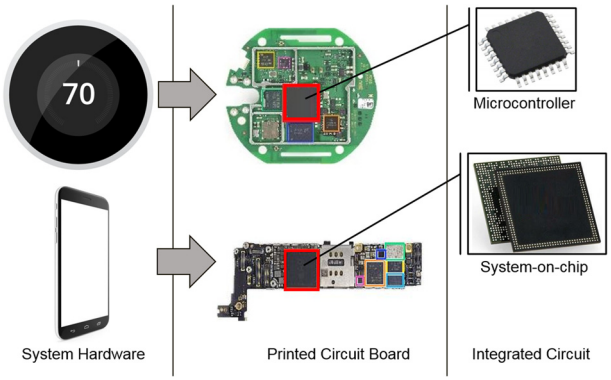

FIGURE 1.4

Three abstraction layers of modern electronic hardware (shown for two example devices).

components (such as PCBs, peripheral devices, and enclosures) that make a system, such as a smart thermostat, or a smartphone. At the next level, we have one or more PCBs, which provide mechanical support and electrical connection to the electronic components that are required to meet the functional and performance requirements of a system. PCBs are typically constructed with multiple layers of an insulating substrate (for example, fiberglass) that allow power and signals to be connected among components using conductive metal (e.g., copper) traces. At the bottom-most layer, we have active components (such as ICs, transistors, and relays), and passive electronic components. Different layers of hardware abstraction bring in diverse security issues, and require commensurate protections. The book covers major security issues and solutions at all levels of hardware abstraction.

1.2.2 TYPES OF ELECTRONIC HARDWARE

The ICs or chips used in a PCB do various tasks, such as signal acquisition, transformation, processing, and transfer. Some of these chips (for example, an encryption or image compression chip) work on digital signals and are called digital ICs, whereas others work on analog or both types of signals, and called analog/mixed-signal (AMS) chips. Examples of the latter type include voltage regulators, power amplifiers, and signal converters. The ICs can also be classified based on their usage model and availability in the market. Application-specific integrated circuits (ASIC) represent a class of ICs, which contain customized functionalities, such as signal processing or security functions, and meet specific performance targets that are not readily available in the market. On the other hand, commercial off-the-shelf (COTS) ICs are the ones, which are already available in the market, often providing flexibility and programmability to support diverse system design needs. These products can be used out-of-the-box, but often needs to be configured for a target application. Examples of COTS components include field programmable gate arrays (FPGA), microcontrollers/processors, and data converters. The distinction between ASIC and COTS is often subtle, and when a chip manufacturer decides to sell its ASICs into the market, they can become "off-the-shelf" to the original equipment manufacturers (OEMs), who build various computing systems using them.

1.3 WHAT IS HARDWARE SECURITY?

Information or data security have remained an issue of paramount concern for system designers and users alike since the beginning of computers and networks. Consequently, protection of systems and networks against various forms of attacks, targeting corruption/leakage of critical information and unauthorized access, have been widely investigated over the years. Information security, primarily based on cryptographic measures, have been analyzed and deployed in a large variety of applications. Software attacks in computer systems have also been extensively analyzed, and a large variety of solutions have been proposed, which include static authentication and dynamic execution monitoring. Study of hardware security, on the other hand, is relatively new, since hardware has been traditionally considered immune to attacks, and hence used as the trust anchor or "root-of-trust" of a system. However, various security vulnerabilities and attacks on hardware have been reported over the last three decades. Earlier, they primarily focused on implementation-dependent vulnerabilities in cryptographic chips leading to information leakage. However, emerging trends in electronic hardware production, such as intellectual-property-based (IP-based) system on chip (SoC) design, and a long and distributed supply chain for manufacturing and distribution of electronic components—leading to reduced control of a chip manufacturer on the design and fabrication steps—have given rise to many growing security concerns. This includes malicious modifications of ICs, also referred to as Hardware Trojan attacks [12], in an untrusted design house or foundry. This is an example of a hardware security issue, which can potentially provide a kill switch to an adversary. Other examples include side-channel attacks, where secret information of a chip can be extracted through measurement and analysis of side-channels, that is, physical signals, such as power, signal propagation delay, and electromagnetic emission; IP piracy and reverse-engineering, counterfeiting, microprobing attacks on ICs, physical tampering of traces or components in PCBs, bus snooping in PCBs, and access to privileged resources through the test/debug infrastructure. They span the entire life-cycle of hardware components, from design to end-of-life, and across all abstraction levels, from chips to PCBs to system. These attacks, associated vulnerabilities and root causes and their countermeasures form the field of hardware security [1,2,10,13,14].

Another important aspect of hardware security relates to the hardware design, implementation, and validation to enable secure and reliable operation of the software stack. It deals with protecting sensitive assets stored in a hardware from malicious software and network, and providing an appropriate level of isolation between secure and insecure data and code, in addition to providing separation between multiple user applications [1]. Two major topics in this area are as follows. (1) Trusted execution environment (TEE), such as ARM's TrustZone, Intel SGX, and Samsung Knox, which protects code and data of an application from other untrusted applications with respect to confidentiality (the ability to observe a data), integrity (the ability to change it), and availability (the ability to access certain data/code by the rightful owner). The confidentiality, integrity, and availability are referred to as CIA requirements. They form three important pillars for secure execution of software on a hardware platform. Establishment of these requirements is enabled by a joint hardware-software mechanism, with hardware providing architectural support for such an isolation, and facilitating effective use of cryptographic functions, and software providing efficient policies and protocols. (2) Protection of security-critical assets in an SoC through appropriate realization of security policies, such as access control and information flow policies, which govern the CIA requirements for these assets. Figure 1.5 depicts these focus areas of the hardware security field.

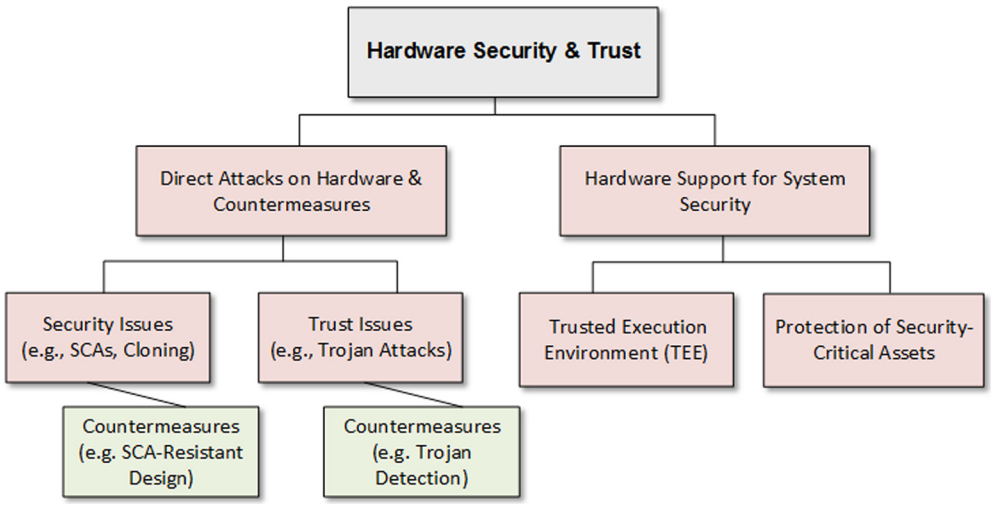

FIGURE 1.5

Scope of hardware security and trust.

1.4 HARDWARE SECURITY VS. HARDWARE TRUST

Hardware security issues arise from its own vulnerability to attacks (e.g., side-channel or Trojan attacks) at different levels (such as, chip or PCB), as well as from lack of robust hardware support for software and system security. On the other hand, hardware trust issues arise from involvement of untrusted entities in the life cycle of a hardware, including untrusted IP or computer-aided design (CAD) tool vendors, and untrusted design, fabrication, test, or distribution facilities. These parties are capable of violating the trustworthiness of a hardware component or system. They can potentially cause deviations from intended functional behavior, performance, or reliability. Trust issues often lead to security concerns, for example, untrusted IP vendor can include malicious implant in a design, which can lead to denial of service (DoS), or information leakage attacks during field operation. However, trust issues can also lead to other incidents, such as poor parametric behavior (for example, reduced performance or energy-efficiency), or degraded reliability, or safety issues. The evolving nature of the global supply chain and the horizontal semiconductor business model are making the hardware trust issues ever more significant. It, in turn, is driving new research and development efforts in trust verification and hardware design for trust assurance.

1.4.1 WHAT CAUSES HARDWARE TRUST ISSUES?

Figure 1.6 shows the major steps in the life cycle of an IC. It starts from a design house creating the functional specifications (e.g., data compression, encryption, or pattern recognition) and parametric specifications (e.g., the operating frequency or standby power) of a design. Next, it goes through a sequence of design and verification steps, where the high-level description of a design (for instance, an architecture level description) is transformed into logic gates, then into a transistor level circuit,

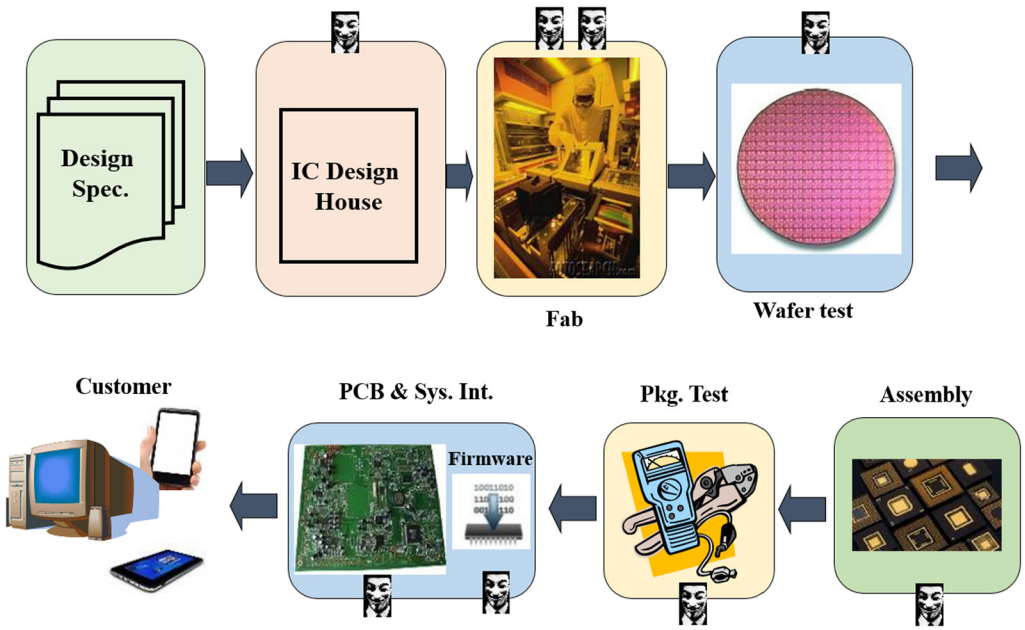

FIGURE 1.6

Major steps in the electronic hardware design and test flow.

and finally, into a physical layout. During this transformation process, a design is verified for correct functional behavior and for performance, power, and other parametric constraints. The layout is then transferred to a fabrication facility, which creates a mask for the layout and then goes through a complex sequence of lithography, etching, and other steps to produce a "wafer", which is typically a circular silicon disk containing a batch of ICs. Each IC in the wafers is then individually tested for certain defects using special test patterns. ICs are referred to as "die" at this stage. These dies are then cut by diamond saw from the wafer and assembled into a package made of ceramic, or other materials. The packaged dies, or ICs, are then tested for compliance with functional and parametric features using another set of test patterns in a manufacturing test facility. This step is vital in the life cycle of an IC, since it ensures that defective chips not meeting functional or parametric specifications are discarded, and do not go into the supply chain. During the early stage of an IC development process, this step is used to identify and debug design bugs (as opposed to manufacturing defects), and information on identified bugs is fed back to the design team in order to incorporate appropriate correction. The testing and debug process for a complex IC is usually facilitated by incorporating specialized structures in a design, which is called design-for-test (DFT) and design-for-debug (DFD) infrastructure, respectively. The primary goal behind inserting these structures is to increase the controllability and observability of internal nodes in a design, which are difficult to access from a fabricated chip. However, as we discuss later, it inherently creates conflict with security goals, which aim to minimize controllability and observability of these nodes, such that an attacker cannot easily access or control internal circuit nodes. For example, direct access to the read/write control for embedded memory in a processor through

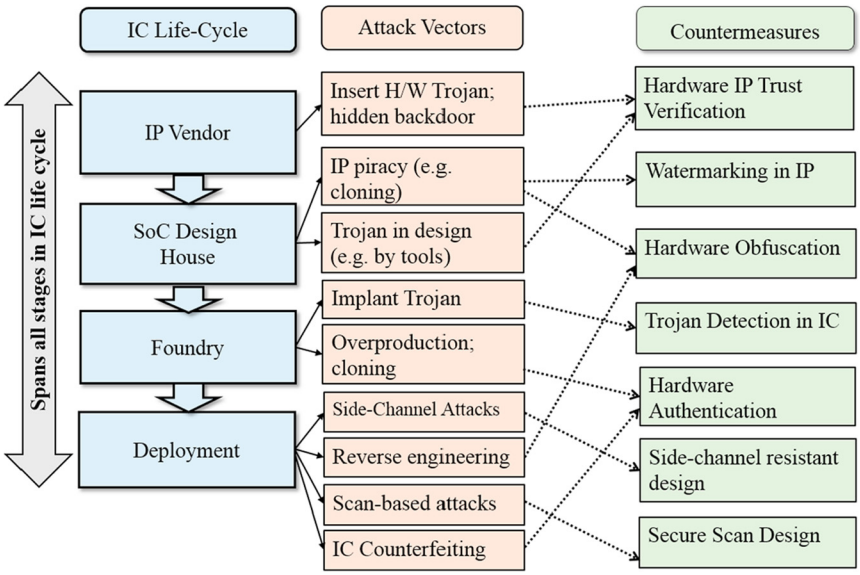

FIGURE 1.7

Attack vectors and countermeasures for each stage in an IC's life span.

the DFT/DFD interface can help an attacker leak, or manipulate sensitive data stored in the protected regions of memory.

The chips that pass manufacturing testing then go into the supply chain for distribution. In current business models, most OEMs acquire these chips from a supply chain, and then integrate them in a PCB, install firmware or configuration bitstream into COTS components, and create a complete system. This long development cycle of hardware involves multiple third-party vendors and facilities. They are often untrusted and globally distributed. In Fig. 1.6, the stages marked by red (medium gray in print version) box are usually untrusted, whereas stages marked with yellow (light gray in print) may or may not be untrusted; the ones marked with green (dark gray in print version) are usually trusted. In the next section, we describe what kind of attacks can be mounted on a hardware in these stages. It is worth noting that, PCB design, fabrication, and test process follow a similar flow, and a horizontal business model—as in the case of IC, where the design and manufacturing companies—is spread around the world to reduce the total production cost. Hence, PCBs are often subject to a similar set of vulnerabilities as ICs.

1.4.2 WHAT SECURITY ISSUES RESULT FROM UNTRUSTED ENTITIES?

Figure 1.7 illustrates some of the key security issues resulting from untrusted design/fabrication/test process for an IC. Connected to the same is our consideration of an SoC life cycle that integrates a number of IPs, typically acquired from third-party IP vendors, into a design that meets functional and performance criteria. These IP vendors are often physically distributed across the globe. Since chip manufacturers do not publish information about their IP sources for business reason, we considered several example SoCs that go into mobile computing platforms (such as cell phones), and created a list

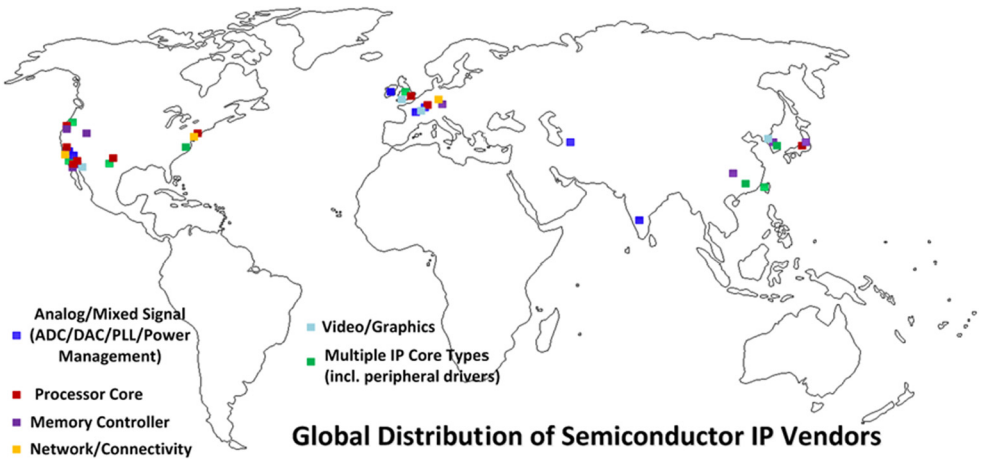

FIGURE 1.8

Long and globally distributed supply chain of hardware IPs makes SoC design increasingly vulnerable to diverse trust/integrity issues.

of common IP blocks, which are integrated into these SoCs [1]. Figure 1.8 shows the map of possible sources of these IPs. Usually, an IP design house specializes in a specific class of IP (for example, memory, communication, or crypto-IP). From this map, it is fair to assume that the IPs used in an SoC are very likely to come from different, and physically distributed third-party IP vendors, which would result in these IPs being untrusted from an SoC designer's point of view. Note that a foundry would have access to the entire unencrypted design file for an SoC, consisting of all IP blocks, the interconnect fabric, and the DFT/DFD structures. While a third-party IP vendor can possibly insert a malicious design component or hardware Trojan, untrusted design, fabrication, and test facilities would have several attack options, such as piracy of a design, reverse engineering, and Trojan implantation. As shown in Fig. 1.7, these security issues can be addressed through targeted design or test solutions, which we will describe later in this book.

1.5 ATTACKS, VULNERABILITIES, AND COUNTERMEASURES

In this section, we briefly introduce the main types of hardware attacks, the threat models for these attacks, the known functional and non-functional vulnerabilities, and the countermeasures that can be taken to protect against these attacks.

1.5.1 ATTACK VECTORS

Attack vectors—as they relate to hardware security—are means or paths for bad actors (attackers) to get access to hardware components for malicious purposes, for example, to compromise it or extract secret assets stored in hardware. Example of hardware attack vectors are side-channel attacks, Trojan attacks, IP piracy, and PCB tampering. Attack vectors enable an attacker to exploit implementation

FIGURE 1.9

Possible attack surfaces in a computing system.

level issues (such as, side-channel attacks and PCB tampering) or take advantage of lack of control on hardware production cycle (such as, Trojan attacks).

1.5.2 ATTACK SURFACE

Attack surface is the sum of all possible security risk exposures. It can also be explained as the aggregate of all known, unknown, and potential vulnerabilities, and controls across all hardware, software, and network components. Tapping into different locations, components, and layers (including hardware/software) of the target system, an attacker can exploit one or more vulnerabilities and mount an attack, for example, extract secret information from a system. Figure 1.9 illustrates major attack surfaces of a smartphone, composed of software, network, data, and hardware components. From the figure, it is evident that the total surface area of a system could be large, and hardware is a critical part of it. In the context of hardware security, attack surfaces define the level of abstraction in which the attacker focuses on launching a hardware attack. Keeping the attack surface as small as possible is a common goal for developing countermeasures. With respect to hardware security, three main attack surfaces are as follows.

Chip Level Attacks: Chips can be targeted for reverse engineering, cloning, malicious insertion, side-channel attacks, and piracy [10,11]. Counterfeit or fake chips can be sold as original units if the attacker can create a copy that has a similar appearance or features as the original. Trojan-infected chips can also find their place in the supply chain, which can pose a threat of unauthorized access, or malfunction. Side-channel attacks can be mounted on a chip with the goal to extract secret information stored inside it. For example, a cryptochip performing encryption with a private key, or a processor

running protected code and/or operating on protected data are both vulnerable to leakage of secret information through this attack.

PCB-Level Attacks: PCBs are common targets for attackers, as they are much easier to reverse-engineer and tamper than ICs. Design information of most modern PCBs can be extracted through relatively simple optical inspection (for example, X-Ray tomography) and efficient signal processing. Primary goals for these attacks are to reverse engineer the PCB, and obtain the schematic of the board to redesign it and create fake units. Attackers may also physically tamper a PCB (for instance, cut a trace or replace a component) to make them leak sensitive information, or bypass DRM protection.

System-Level Attacks: Complex attacks involving the interaction of hardware-software components can be mounted on the system. By directly focusing on the most vulnerable parts in a system, such as DFT infrastructure at PCB level (for example, JTAG) and memory modules, attackers may be able to compromise the system's security by gaining unauthorized control and access to sensitive data.

1.5.3 SECURITY MODEL

Attacks on hardware systems can take many forms. An attacker's capabilities, physical or remote access of the system, and assumptions of system design and usage scenarios play essential roles in the techniques that can be used to launch an attack. In order to describe a security issue or solution, it is important to unambiguously describe the corresponding security model. A security model should have two components: (1) Threat Model, which describes the threats including, the purpose and mechanism of an attack; and (2) Trust Model, which describes the trusted parties or components. In order to describe the security issues arising from malicious implants in third-party IPs, the threat model needs to describe the objective of the attackers, for example, to leak secret from an SoC or to disrupt its functional behavior; and the way the attack is mounted, for instance, through the insertion of a Trojan that triggers malicious memory write operation under a rare internal condition. The trust model needs to describe which parties are trusted, for example, the SoC designer and CAD tools are trusted in this case.

1.5.4 VULNERABILITIES

Vulnerabilities refer to weakness in hardware architecture, implementation, or design/test process, which can be exploited by an attacker to mount an attack. These weaknesses can either be functional or nonfunctional, and they vary based on the nature of a system and its usage scenarios. A typical attack consists of an identification of one or more vulnerabilities, followed by exploiting them for a successful attack. Identification of vulnerabilities is usually the hardest step in the attack process. Following is a description of some typical vulnerabilities in hardware systems:

Functional Bug: Most vulnerabilities are caused by functional bugs and poor design/testing practices. They include weak cryptographic hardware implementation and inadequate protection of assets in an SOC. Attackers may find these vulnerabilities by analyzing the functionality of a system for different input conditions to look for any abnormal behaviors. Additionally, vulnerabilities may be discovered accidentally, which makes it easier for an attacker to perform malicious activities using these newly discovered issues in the system.

Side-Channel Bug: These bugs represent implementation-level issues that leak critical information stored inside a hardware component (for example processors or cryptochips) through different forms of side-channels [4]. Attackers may find these vulnerabilities by analyzing the side-channel signals

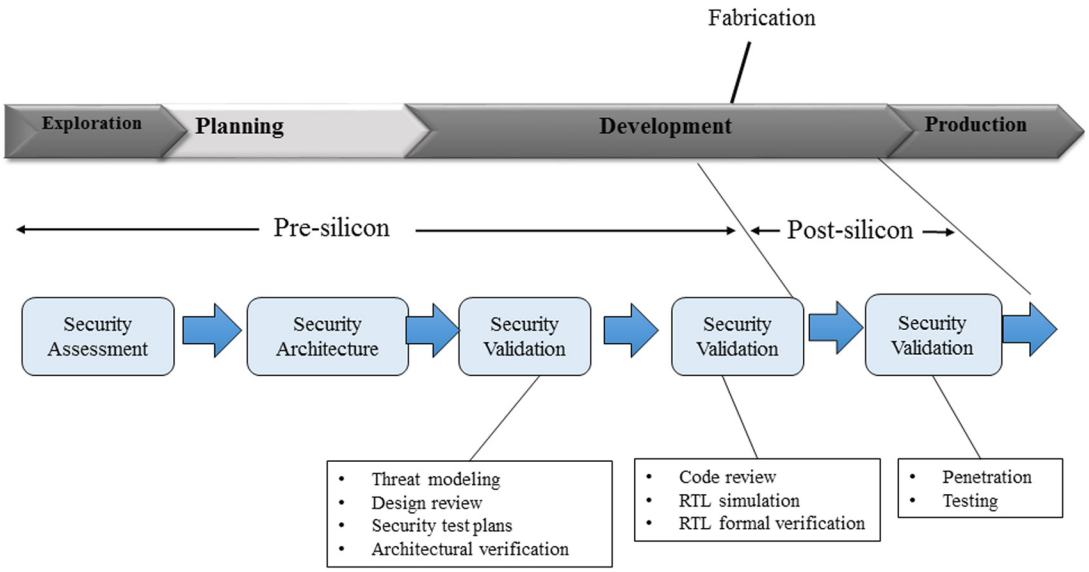

FIGURE 1.10

State of the practice in security design and validation along the life cycle of a system on chip.

during operation of a hardware component. Many powerful attacks based on side-channel bugs rely on statistical methods to analyze the measured traces of a side-channel parameter [2]. Criticality of a side-channel bug depends on the amount of information leakage through a side channel.

Test/Debug infrastructure: Most hardware systems provide a reasonable level of testability and debuggability, which enable designers and test engineers to verify the correctness of operation. They also provide means to study internal operations and processes running in a hardware, which are essential for debugging a hardware. These infrastructures, however, can be misused by attackers, where extraction of sensitive information or unwanted control of a system can be possible using the test/debug features.

Access control or information-flow issues: In some cases, a system may not distinguish between authorized and unauthorized users. This vulnerability may give an attacker access to secret assets and functionality that can be misused or leveraged. Moreover, an intelligent adversary can monitor the information flow during system operation to decipher security-critical information, such as, control flow of a program and memory address of a protected region from a hardware.

1.5.5 COUNTERMEASURES

As hardware attacks have emerged in the past years, countermeasures to mitigate them have also been reported. Countermeasures can either be employed at design or test time. Figure 1.10 shows the current state of the practice in the industry for SoCs in terms of: (a) incorporating security measures in a design (referred to as "security design"), and (b) verifying that these measures protect a system against known attacks (referred to as "security validation"). SoC manufacturing flow can be viewed as consisting of

four conceptual stages: (1) exploration, (2) planning, (3) development, and (4) production. The first two stages and part of the development stage form the pre-silicon part of SoC life cycle, which consists of exploring the design space, architecture definition, and then deriving a design that meets design targets. Part of the development stage, followed by the production of SoCs, form the post-silicon part of the SoCs' life, which consists of verifying and fabricating the chips. Security assessment is performed during the exploration stage, which identifies the assets in an SoC, possible attacks on them, and requirements for secure execution of software, when applicable. This step ends up creating a set of security requirements. Next, an architecture is defined (referred to as "security architecture") to address these requirements, which includes protection of test/debug resources against malicious access, and safeguarding cryptographic keys, protected memory regions, and configuration bits. Once the architecture is defined and the design is gradually created, pre-silicon-security validation is performed to make sure the architecture and its implementation adequately fulfill the security requirements. Similar security validation is performed after chips are fabricated (referred to as "post-silicon security validation") to ensure that the manufactured chips do not have security vulnerabilities and, hence, are protected against known attacks. Both pre- and post-silicon security validation come in various forms, which vary in terms of coverage of security vulnerabilities, the resulting confidence, and the scalability of the approach to large designs. These techniques include code review and formal verification during pre-silicon validation, fuzzing, and penetration testing during post-silicon validation [16].

Design solutions: Design-for-security (DfS) practices have emerged as powerful countermeasures. DfS offers effective low-overhead design solutions that can provide active or passive defense against various attacks. DfS techniques, such as obfuscation [6], use of reliable security primitives, side-channel resistance (for example, masking and hiding techniques), and hardening schemes for Trojan insertion, can reliably protect against many major attack vectors. Likewise, SoC security architecture that is resilient against software attacks has been a significant aspect of SoC platform security.

Test and verification solutions: Test and verification techniques have constituted a major category of protection approaches against the diverse security and trust issues. Both pre-silicon verification—functional as well as formal—and post-silicon manufacturing testing have been considered as mechanisms to identify security vulnerabilities and trust issues for chips, PCBs, and systems. The book covers various DfS and test/verification solutions, which are developed to protect hardware against many vulnerabilities.

1.6 CONFLICT BETWEEN SECURITY AND TEST/DEBUG

Security and test/debug of an SoC often impose conflicting design requirements during its design phase. Post-manufacturing testing and debug using DFT, for example, scan chain, and DFD structures constitute some of the important activities in a SoC lifecycle. Effective debug demands internal signals of IP blocks to be observable during execution in silicon. However, security constraints often cause severe restrictions to internal signal observability, thus making debugging a challenge. These constraints arise from the need to protect many critical assets, such as, locks for high-assurance modules, encryption keys, and firmware. While these security assets themselves are difficult to observe during debugging, they also create observability challenge for other signals, for example, signals from an IP containing low-security assets that need to be routed through an IP block with a high-security asset.

Unfortunately, in current industrial practice, this problem is difficult to address. First, it lacks formal centralized control on security assets, since they are determined per-IP basis. Second, debug requirements are usually not considered during the integration of security assets, which often leads to the discovery of the debug issues very late during actual debug with silicon execution. Fixing the problem at that point may require a silicon "respin", that is, design correction followed by re-fabrication, which is expensive and often an unacceptably long process. Hence, there is a growing emphasis to develop hardware architecture, which ensures the security of DFT and DFD infrastructure, while ensuring their desired role in helping with SoC test/debug process.

1.7 EVOLUTION OF HARDWARE SECURITY: A BRIEF HISTORICAL PERSPECTIVE

Over the past three decades, the field of hardware security has been evolving rapidly with the discovery of many vulnerabilities and attacks on hardware. Figure 1.11 provides a brief timeline for the evolution of hardware security. Before 1996, there were only sporadic instances of hardware IP piracy, primarily cloning of ICs, leading to the development of some IP watermarking and other anti-piracy techniques. In 1996, a groundbreaking hardware attack was introduced in the form of timing analysis attack [3], an attack which aims to extract information from a cryptographic hardware on the basis of a systematic analysis of computation time for different operations. In 1997, fault injection analysis was reported as an attack vector that can lead to compromising the security of a system [7]. The attack focuses on applying environmental stress to the system in order to force it to leak sensitive data. The first power analysis based side-channel attack was introduced in 1999 [2]; it focused on analyzing the power dissipations at runtime to retrieve secrets from a cryptochip.

In 2005, there were reports on production and supply of counterfeit ICs, including cloned and recycled chips, which created major security and trust concerns. The concept of hardware Trojans was introduced in 2007 [12], which unveiled the possibility of inserting malicious circuits in a hardware design with the aim to disrupt normal functional behavior, leak sensitive information, grant unauthorized control, or degrade the performance of the system. Some recent hardware vulnerabilities that have received significant attention from industry and academic community includes "Meltdown" and "Spectre" [9]; they exploit implementation-dependent side-channel vulnerabilities in modern processors to

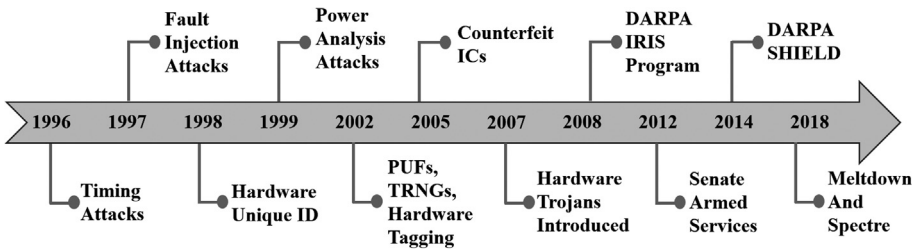

FIGURE 1.11

The evolution of hardware security over the past decades.

access private data from a computer, such as user passwords. These vulnerabilities have been discovered and reported by different processor manufacturers, who have introduced software fixes for them.

Similar to the realm of software security, countermeasures for hardware attacks have been developed in a reactive manner. Over the years, many design and test solutions have evolved to mitigate known attacks. The idea of hardware tagging was introduced in 1998, where every IC instance was assigned with a unique ID. Hardware security primitives, such as physical unclonable functions (PUFs) and true random number generators (TRNGs) were introduced in early 2000 to improve the level of protection against hardware attacks [5,15]. The United States Department of Defense introduced several sponsored research programs to facilitate growth in hardware security solutions. In 2008, DARPA introduced the Integrity and Reliability of Integrated Circuits (IRIS) program to develop techniques for hardware integrity and reliability assurance through destructive and nondestructive analysis. In 2012, a report published by the senate armed services showed that a set of counterfeit devices was discovered in different branches of the US Air Force [8], accentuating the gravity of the problem. The total number of these counterfeits exceeded 1 million, and the investigation concluded with an amendment that enforces counterfeit-avoidance practices. The Supply Chain Hardware Integrity for Electronics Defense (SHIELD) program was introduced by DARPA in 2014 to develop technology to trace and track electronic components—PCB to chip to small passive components—as they move through the supply chain. Over the past decade, many such efforts by both government and industry to enable secure and trusted hardware platform have been observed with more to come in near future.

1.8 BIRD'S EYE VIEW

Table 1.1 provides a bird's-eye view on major hardware security issues and countermeasures, which we have covered in this book. For each attack, it provides information on the adversary, attack surface, and attack objective; whereas for a countermeasure, it lists the stage of hardware lifecycle when it is applied, the goal, and the associated overhead. This table is expected to serve as a quick reference for the readers to some of the key concepts presented in the book.

1.9 HANDS-ON APPROACH

We have included hands-on experiments for several major hardware security topics in this book. We believe a practical learning component is crucial in understanding the diverse security vulnerabilities and the defense mechanisms in a complex system. To do the experiments, we have custom-designed an easy-to-understand, flexible, and ethically "hackable" hardware module, in particular, a printed circuit board (PCB) with basic building blocks that can emulate a computer system and create a network of connected devices. It is called "HaHa", that is Hardware Hacking module. Appendix A provides a detailed description of the HaHa board and associated components. Relevant chapters of the book include a short description of the experiments that can be performed to better understand the topic of the chapters. The experiments, we also hope, would help to stimulate interest in students to further investigate the security issues, and to explore effective countermeasures. In addition to the board, the hands-on experiment platform includes corresponding software modules, and well-written instructions to realize diverse security attacks in this platform, all of which are available as companion materials in the book's own website.

Table 1.1 Bird's-eye view of the hardware attacks & countermeasures

ATTACKS					
Type of Attack	**What it is**	**Adversary**	**Goal**	**Life-cycle stages**	**Chapter #**
Hardware Trojan Attacks	Malicious design modification (in chip or PCB)	Untrusted foundry, untrusted IP Vendor, untrusted CAD tool, untrusted design facilities	• Cause malfunction • Degrade reliability • Leak secret info	• Design • Fabrication	Chapter 5
IP Piracy	Piracy of the IP by unauthorized entity	Untrusted SoC Designer, untrusted foundry	• Produce unauthorized copy of the design • Use an IP outside authorized use cases	• Design • Fabrication	Chapter 7
Physical Attacks	Causing physical change to hardware or modifying operating condition to produce various malicious impacts	End user, bad actor with physical access	• Impact functional behavior • Leak information • Cause denial of service	• In field	Chapter 11
Mod-chip Attack	Alteration of PCB to bypass restrictions imposed by system designer	End user	• Bypass security rules imposed through PCB	• In field	Chapter 11
Side-Channel Attacks	Observing parametric behaviors (i.e., power, timing, EM) to leak secret information	End user, bad actor with physical access	• Leak secret information being processed inside the hardware	• In field	Chapter 8
Scan-based Attacks	Leveraging DFT circuits to facilitate side-channel attack	End user, bad actor with physical access	• Leak secret information being processed inside the hardware	• In field • Test-time	Chapter 9
Microprobing	Using microscopic needles to probe internal wires of a chip	End user, bad actor with physical access	• Leak secret information residing inside the chip	• In field	Chapter 10
Reverse Engineering	Process of extracting the hardware design	Design house, foundry, end user	• Extract design details of the hardware	• Fabrication • In field	Chapter 7

(continued on next page)

Table 1.1 (*continued*)

COUNTERMEASURES					
Type of Countermeasure	**What it is**	**Parties involved**	**Goal**	**Life-cycle stages**	**Chapter #**
Trust Verification	Verifying the design for potential vulnerabilities to confidentiality, integrity, and availability	• Verification engineer	• Provide assurance against known threats	• Pre-silicon verification • Post-silicon validation	Chapter 5
Hardware Security Primitives (PUFs, TRNGs)	Providing security features to support supply chain protocols	• IP integrator • Value added reseller (for enrollment)	• Authentication • Key generation	• Throughout IC supply chain	Chapter 12
Hardware Obfuscation	Obfuscating the original design to prevent piracy and reverse engineering	• Design house • IP integrator	• Prevent piracy • Reverse engineering • Prevent Trojan insertion	• Design-time	Chapter 14
Masking & Hiding	Design solutions to protect against side-channel attacks	• Design house	To prevent side-channel attacks by reducing leakage or adding noise	• Design-time	Chapter 8
Security Architecture	Enable design-for-security solution to prevent potential and emerging security vulnerabilities	• Design house • IP integrator	Address confidentiality, integrity, and availability issues with design-time solution	• Design-time	Chapter 13
Security Validation	Assessment of security requirements	• Verification and validation engineer	Ensure data integrity, authentication, privacy requirements, access control policies	• Pre-silicon verification • Post-silicon validation	Chapter 16

1.10 EXERCISES

1.10.1 TRUE/FALSE QUESTIONS

1. Hardware is not considered as the “root-of-trust” for system security.
2. Hardware security should not matter if a strong software tool is used to protect user’s data.
3. Hardware contains different forms of assets that can be accessed by bad actors.

4. Meltdown and Spectre are two newly discovered vulnerabilities found in most modern processors.
5. Hardware development lifecycle involves a number of untrusted entities.
6. Hardware trust issues do not lead to any security issue.
7. Side-channel attacks are attack vectors that exploit implementation-level weakness.
8. Test and debug features in a hardware often represent a conflict with security objectives.
9. A functional bug can be exploited by an attacker for extracting assets in a SoC.
10. Verification solutions can protect against several hardware security issues.

1.10.2 SHORT-ANSWER TYPE QUESTIONS

1. Describe different levels of abstraction of electronic hardware.
2. State the differences: (1) general-purpose systems vs. embedded systems, (2) ASIC vs. COTS.
3. Describe two major areas of focus for hardware security.
4. What are the hardware trust issues, and how do they impact the security of a computing system?
5. What are the differences between functional and side-channel bugs?
6. Why and how do security and test/debug requirements conflict?
7. Provide examples of some security assets inside SoCs.

1.10.3 LONG-ANSWER TYPE QUESTIONS

1. Describe different aspects of a system's security, and briefly discuss their relative impact.
2. Explain the current state of practice in the security design of and verification process for SoCs.
3. Describe the major steps of the electronic hardware design and test flow, and discuss the security issues in each stage.
4. What are the different attack surfaces for a computing system (say, a smartphone), and for the hardware components inside it?
5. Describe different types of security vulnerabilities in hardware.

REFERENCES

[1] S. Ray, E. Peeters, M.M. Tehranipoor, S. Bhunia, System-on-chip platform security assurance: architecture and validation, Proceedings of the IEEE 106 (1) (2018) 21–37.
[2] P. Kocher, J. Jaffe, B. Jun, Differential power analysis, in: CRYPTO, 1999.
[3] P. Kocher, Timing attacks on implementations of Die–Hellman, RSA, DSS, and other systems, in: CRYPTO, 1996.
[4] F. Koeune, F.X. Standaert, A tutorial on physical security and side-channel attacks, in: Foundations of Security Analysis and Design III, 2005, pp. 78–108.
[5] M. Barbareschi, P. Bagnasco, A. Mazzeo, Authenticating IoT devices with physically unclonable functions models, in: 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 2015, pp. 563–567.
[6] A. Vijayakumar, V.C. Patil, D.E. Holcomb, C. Paar, S. Kundu, Physical design obfuscation of hardware: a comprehensive investigation of device and logic-level technique, IEEE Transactions on Information Forensics and Security (2017) 64–77.
[7] J. Voas, Fault injection for the masses, Computer 30 (1997) 129–130.
[8] U.S. Senate Committee on Armed Services, Inquiry into counterfeit electronic parts in the Department of Defense supply chain, 2012.
[9] Meltdown and Spectre: Here's what Intel, Apple, Microsoft, others are doing about it. https://arstechnica.com/gadgets/2018/01/meltdown-and-spectre-heres-what-intel-apple-microsoft-others-are-doing-about-it/.
[10] M. Tehranipoor, U. Guin, D. Forte, Counterfeit integrated circuits, Counterfeit Integrated Circuits (2015) 15–36.

[11] R. Torrance, D. James, The State-of-the-Art in Semiconductor Reverse Engineering, ACM/EDAC/IEEE Design Automation Conference (DAC) (2011) 333–338.
[12] M. Tehranipoor, F. Koushanfar, A Survey of Hardware Trojan Taxonomy and Detection, IEEE Design and Test of Computers (2010) 10–25.
[13] Y. Alkabani, F. Koushanfar, Active Hardware Metering for Intellectual Property Protection and Security, Proceedings of 16th USENIX Security Symposium on USENIX Security (2007) 291–306.
[14] G. Qu, F. Koushanfar, Hardware Metering, Proceedings of the 38th annual Design Automation (2001) 490–493.
[15] R. Pappu, B. Recht, J. Taylor, N. Gershenfeld, Physical One-Way Functions, Science (2002) 2026–2030.
[16] F. Wang, Formal Verification of Timed Systems:A Survey and Perspective, Proceedings of the IEEE (2004) 1283–1305.