Introduction to Hardware Security

Yier Jin

Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32765, USA; E-Mail: yier.jin@eecs.ucf.edu; Tel.: +1-407-823-5321; Fax: +1-407-823-5835

Academic Editor: Dhananjay S. Phatak

Received: 3 August 2015 / Accepted: 5 October 2015 / Published: 13 October 2015

Abstract: Hardware security has become a hot topic recently with more and more researchers from related research domains joining this area. However, the understanding of hardware security is often mixed with cybersecurity and cryptography, especially cryptographic hardware. For the same reason, the research scope of hardware security has never been clearly defined. To help researchers who have recently joined in this area better understand the challenges and tasks within the hardware security domain and to help both academia and industry investigate countermeasures and solutions to solve hardware security problems, we will introduce the key concepts of hardware security as well as its relations to related research topics in this survey paper. Emerging hardware security topics will also be clearly depicted through which the future trend will be elaborated, making this survey paper a good reference for the continuing research efforts in this area.

Keywords: hardware security; hardware trojan; proof-carrying hardware; trusted hardware platform

1. Introduction

Hardware has long been viewed as a trusted party supporting the whole computer system and is often treated as an abstract layer running instructions passed from the software layer. Therefore, hardware-related security research is often referred to hardware implementations of cryptographic algorithms where hardware is used to improve the calculation performance and efficiency for cryptographic applications [1]. Hardware copyright protections are also categorized as hardware related security research where watermarking is widely used to solve the copyright issues [2]. However, researchers from these areas do not consider the protection on the hardware itself. For a long time, cybersecurity researchers believed that the integrated circuit (IC) supply chain was well-protected with
high barriers such that attackers could not easily compromise the fabricated chips. With the high cost of cutting-edge foundries and increasing design complexity of modern system-on-chip (SoC) platforms, the IC supply chain, which was once located in one country or even in one company, has been spread around the globe. Following this trend, third-party resources in hardware circuit designs, mostly in the format of third-party fabrication services and third-party soft/hard IP cores for SoC development, are prevalingly used in modern circuit designs and fabrications. The availability of those resources largely alleviates the design workload, lowers the fabrication cost, and shortens the time-to-market (TTM). However, the heavy reliance on third-party resources/services also breeds security concerns and invalidates the illusion that attackers cannot easily access the isolated integrated circuit (IC) supply chain. For example, a malicious foundry may insert hardware Trojans into fabricated chips. The delivered IP cores may contain malicious logic and/or design flaws which could be exploited by attackers after the IP cores are integrated into SoC platforms.

The concept of hardware security was formally introduced after the emergence of hardware Trojans and the following countermeasures to mitigate or prevent this kind of threat. Hardware security was a term which originally referred to hardware Trojan designs, categorization, detection, and isolation where the untrusted foundries were treated as the main threats. As a result, the developed hardware Trojan detection methods often focus on the post-silicon phase with emphasis on the security enhancement of existing testing methods [3–18]. Given the fact that third-party IP cores may be another attack vector for malicious logic insertion, the protection of pre-synthesis designs becomes equally important. Following this request, pre-silicon circuit protection approaches have also been developed [19–24].

Besides the scope of hardware Trojan detection, the concept of hardware security has been expanded from testing solutions to formal verification approaches. While formal methods have been widely used in software program security assurance, they have also been proven to be effective in security verification on hardware code, which is often written in a hardware description language (HDL) [25–27]. The development of these methods help provide high-level security assurance to hardware designs even in the circumstance that attackers may have the access to the original designs. These formal methods also help overcome the limitations of the requirement of golden models within many other hardware Trojan detection methods. Nevertheless, the construction of security properties becomes an open question that hardware security researchers are trying to solve.

The evolution of hardware security research recently moved away from the hardware Trojan detection and now leans towards trustworthy hardware development for the construction of the root-of-trust. The intrinsic properties of hardware devices which have a negative impact on circuit performance are leveraged for security applications. One leading example is the development of physical-unclonable functions (PUFs) which rely on device process variation to generate chip-specific fingerprints in the format of challenge-response pairs. Looking beyond MOSFETs, researchers are investigating the use of emerging transistors, such as spin-transfer torque (STT) device, memristor, and spontronic domain wall, leveraging their special properties for hardware security applications.

Another trend in the hardware security area is the development of security-enhanced hardware infrastructure for device protection. A new hardware infrastructure that integrates security, protection, and authentication into a dedicated tool-chain is imminently essential. Various security-enhanced architectures are under development [28–35]. This survey will introduce hardware-level enhancements
that provide high-level security and emphasize usability and protection extending to all architectural layers. The new trend relies on a re-evaluation of hardware functionality, performance, reliability, design, and verification from a security-enhanced perspective. The goal is to provide a fully operational software and hardware platform that ensures secure design, manufacturing, and deployment of modern computer systems. Specific focus will be given to security-enhanced hardware processors and SoC designs that support secure boot and access control so that both hardware- and firmware-level attacks can be prevented.

Meanwhile, given the fast expansion of the research topics within the hardware security area, more and more researchers from related areas have been joining this area recently. Therefore, to help researchers newly joined in this area better understand the challenges and tasks within hardware security domain and to help them investigate countermeasures and solutions to solve hardware security problems, this survey paper will act as a concise guide introducing the history of hardware security as well as emerging topics and challenges. Interested readers are also encouraged to attend the hardware security dedicated conference, IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) [36], and the books introducing related topics [37].

The rest of this paper is organized as follows: Section 2 will introduce work on hardware Trojan detection and prevention methods. Section 3 elaborates researchers’ efforts in leveraging formal methods for hardware IP protection and trustworthiness verification. Section 4 will depict a new hardware security threat on device counterfeiting where state-of-the-art solutions will also be discussed. While physical-unclonable functions (PUFs) have been widely introduced in other literature, a brief discussion of the new PUF design trends will be presented in Section 5. The future trend in hardware security research will be discussed in Section 6 about emerging devices in hardware security applications and in Section 7 about hardware-assisted computer security. Final conclusions are drawn in Section 8.

2. Hardware Trojan Detection

2.1. Pre-Deployment Hardware Trojan Detection

Unlike software viruses and software Trojans, hardware Trojans cannot be easily eliminated through firmware updating and therefore, are more harmful to computer systems. Hardware Trojans are designed by attackers to add unwanted functionality to the design. There is not a standard procedure to design hardware Trojans, as its design is reliant upon the attackers goals and available resources. Despite this, hardware security researchers have categorized different Trojans. For example, authors in [7] divided Trojans into implicit payload and explicit payload based on the Trojan’s activities when triggered. A more detailed Trojan taxonomy can be found in [38,39]. Various Trojan designs were also proposed based on the stealth of hardware Trojans and the impact they may cause [40–43]. While most of these Trojans are inserted at the register transfer level (RTL), Trojan insertion is also possible through dopant manipulation [44].

As existing testing methods fall short in detecting malicious logic, dedicated hardware Trojan detection methods and trusted integrated circuit design have been developed in recent years [45]. A large body of hardware Trojan detection and prevention methods have been proposed, which can be divided
into four main categories: (i) enhanced functional testing [46]; (ii) side-channel fingerprinting [47]; (iii) Trojan prevention; and (iv) circuit hardening [6].

The enhanced functional testing method is based on the idea that hardware Trojans often rely on rarely triggered events. Therefore, researchers propose to either include those rare events in the testing patterns to trigger the Trojan during the testing stage [5], or analyze all rare events at the gate-level netlists to identify suspicious nodes which may serve as triggers [19]. This method suffers from the limitation that no standard definition for rare events exists, leaving a huge gap between standard testing patterns and rare event patterns.

Side-channel fingerprinting is another popular solution. Even though a hardware Trojan cannot be triggered easily during the testing stage, it can evade functional testing, the inserted Trojan has to alter the parametric profile of a contaminated circuit [38,48,49]. The effectiveness of this method relies on the capability to differentiate side-channel signals of Trojan-infected circuits from Trojan-free circuits. Thus, advanced data analysis methods are utilized to help generate side-channel fingerprints by eliminating the increasing process variation and measurement noise [14,47]. Various side-channel parameters and their combinations are chosen for fingerprint generation and Trojan detection, which include global power traces [47], local power traces [3,4], path delays [7,50], etc. Side-channel fingerprinting-based Trojan detection has been widely used for its non-intrusive properties, but this method is developed based on the assumption that a golden model should be available for comparison, which is not often possible.

Trojan prevention and circuit hardening techniques try to modify the circuit’s structure with extra logic either to eliminate rare/suspicious events [6,22], or to make the target circuit more sensitive to malicious modifications [21]. These methods are often combined with other Trojan detection methods to increase the detection accuracy or to lower testing cost. Even though circuit co-design techniques are used to lower the impact of additional logic in the target design, the extra protection logic will still impact circuit performance. Furthermore, the hardening structure can itself be a target of hardware Trojans [51].

2.2. Post-Deployment Hardware Trojan Detection

Although existing Trojan detection methods have hitherto been successful in detecting certain hardware Trojans, the scope of detection is limited because they rely on over-simplified, sometimes erroneous assumptions including:

- hardware Trojan designers use traditional and simple circuit structures which limit their functionality;
- hardware Trojan designers attempt to occupy negligible on-chip area in order to mask the Trojan profile from the overall side-channel profile;
- golden models are available for the circuit-under-test that can be used to detect side-channel profile deviations;
- attackers will only target digital circuits because analog/RF circuits are more sensitive to modifications.

These assumptions are widely embraced by the hardware security community such that they are the principal guidelines for the development of state-of-the-art hardware Trojan detection/prevention
methods and research. Unfortunately, these assumptions are proving invalid, and it is becoming more
apparent that the problem of hardware security and hardware Trojan detection is more complicated and
broader than what we once imagined. In particular:

- hardware Trojans, similar to the progression seen in modern circuit design, can utilize advanced
design techniques to improve inconspicuousness without sacrificing functionality [40,42];
- enhanced Trojan designs can use significant chip space and still remain hidden with respect to the
overall side-channel profile [52];
- golden models are not always available for integrated systems containing 3rd-party
resources [41,53];
- analog/RF circuits are equally vulnerable to hardware Trojan attacks [48,54,55].

These aforementioned findings invalidate many previously proposed Trojan detection/prevention
methods leaving integrated systems, both in the digital and analog/RF domains, vulnerable to hardware
Trojan attacks.

Therefore, researchers started to look into post-deployment methods leveraging post-deployment
side-channel fingerprinting and on-chip equivalence checking. The key idea here was that stealthy
hardware Trojans may easily evade detection methods during the testing stage but, if triggered, will have
a large impact on side-channel fingerprinting or on circuit functionality. The first post-deployment trust
evaluation structure has been proposed [54]. However, this trusted evaluation process has the limitation
that it will only be triggered externally through primary inputs halting the normal operation and leaving
plenty of time for attackers to trigger and mute the Trojans during the testing intervals. To overcome
the shortage of this method, a real-time trust evaluation structure is proposed that can constantly monitor
the operational status of the target circuit and report circuit abnormalities instantly [52]. Concurrent
on-chip parity checking leveraging on resistive RAM is also developed for run-time hardware Trojan
detection. [56].

3. Formal Verification

Besides circuit level protection methods, formal verification is also widely used for pre-silicon
functionality and security validation. Among all formal methods for security verification, theorem
proving is a representative solution which can provide high level protection to the underlying designs.
However, this method also suffers from high computation complexity, a tedious proof construction
process, and the lack of automation tools. In this section, we will introduce a newly proposed
proof-carrying hardware (PCH) framework on hardware IP cores [57]. We will also introduce a
SAT-solver-based formal protection method for IP trustworthiness validation.

3.1. Proof-Carrying Hardware

Proof-Carrying Hardware is an approach for ensuring trustworthiness of hardware [24,25,58,59]. The
PCH method is inspired from the proof-carrying code (PCC), which was proposed by G. Necula [60].
Using the PCC mechanism, untrusted software developers/vendors certify their software code. During
the certification process, software vendors develop safety proofs for the safety policies provided by
software customers. The vendor then provides the user with a PCC binary file, which includes the formal proof of the safety properties encoded with the executable code of the software. The customer becomes assured of the safety of the software code by quickly validating the PCC binary file in a proof checker. Efficiency of this approach in reducing validation time at the customer end led to its adoption in different applications. Despite the advantages, the PCC approach demanded a large trusted computing base (TCB). This limitation of PCC was overcome in foundational PCC (FPCC), which uses foundation logic to express operational semantics of different assembly language instructions and implements the logic in a logical framework [61–63]. However, the FPCC method increased the proof development effort by many-fold. Subsequently, the Certified Assembly Programming (CAP) language was developed, which used Hoare-logic style reasoning for proof construction [64,65].

Following the concept of PCC framework, authors in [24,59] proposed a preliminary Proof-Carrying Hardware (PCH) for dynamically reconfigurable hardware platforms. The key idea here is to leverage runtime combinational equivalences checking between the design specification and the design implementation on reconfigurable platforms [59]. This method, though named PCH, is in fact a SAT solver-based combinational equivalence checking with the only exception that the resolution proof traces are treated as proofs for the functional equivalence. In this framework, the applied safety policy has nothing to do with the security property. Rather, the safety policy agreed by both bitstream providers and IP users is to make sure that both sides follow the same bitstream formats, the conjunctive normal form to represent combinational functions, and the propositional calculus for proof construction and verification. A more detailed introduction to the SAT solver based proof-carrying framework on configurable platforms was then introduced in [66]. The hardware trust and threat model as well as experimental setup are elaborated on in [23]. Although the authors claimed that the proposed PCH concept would have more potential applications based on other safety properties, the proof-of-concept demonstration still relied on runtime combinational equivalence checking (CEC) to only check the functional equivalence between specifications and implementations.

In order to apply the proof-carrying method to protect general pre-synthesis register-transfer level (RTL) IP cores other than the post-synthesis FPGA bitstreams, a new PCH framework was developed in [25,58] which was built upon the CAP’s construction and application. Similar to FPCC and CAP framework, the new PCH framework uses the Coq functional language for proof construction and leverages the Coq platform for automatic proof validation [67]. The usage of the Coq platform ensures that both IP vendors and IP consumers will implement the same deductive rules and helps simplify the proof validation procedure. However, the Coq platform cannot recognize the commercial hardware description languages (HDLs) directly. To solve this problem, a formal temporal logic model was developed to represent hardware circuits in Coq environment. Conversion rules were also developed to help convert the commercial IP cores from HDLs into the Coq-based formal logic upon which the security property theorems and their proofs can be constructed [25]. Based on this PCH framework, a new trusted IP acquisition and delivery protocol are proposed [57]. From this framework, IP consumers will provide both functional specification and a security property set to IP vendors. IP vendors will then prepare the HDL code based on the functional specification. The circuit will then be converted into a formal representation from HDL code for security theorem generation and proof construction. The HDL code and the formal theorem-proof pairs will be combined into the trusted bundle to be delivered. Upon
receiving the trusted bundle, IP consumers will use the same conversion rules to generate the formal circuit representation first. The converted code, combined with the formal theorems and proofs, will be loaded into the proof checker. Through an automatic property checking process, IP consumers can quickly validate the trustworthiness of delivered IP cores. Since the computation workload has been shifted from IP consumers to IP vendors, the PCH framework becomes a very attractive method for trusted IP validation. Further, since the automatic proof checking process is fast, IP consumers can check the IP cores every time they want to use the design, making it difficult for even internal attackers to insert malicious logic in the design.

Within the PCH framework defined in [25], the most important component is the set of security properties. A complete set of properties will enhance trustworthiness of the IP core by detecting malicious logic if present in the IP core. However, the PCH methodology and the trusted IP transaction process of [25] does not specify details of security properties for individual designs. Since different hardware IP cores often share similar security properties, in order to lower the design cost of the PCH framework and reuse previously developed theorem-proof pairs, a property library managing different types of security properties in a centralized manner would be a desirable solution. Any given hardware design can pick a certain set of security properties from the property library rather than developing all properties from scratch. The selected set of properties can render many modes of attack significantly more difficult to implement and ensure the trustworthiness of the delivered IP cores. The security property library can be expanded as desired to protect IP cores against different types of hardware Trojan attacks. Serving as the first step toward constructing this library, data secrecy properties are chosen as prevailing properties for most hardware IP cores [20,26]. The key idea of the data secrecy properties is to track the internal information flow and, relying on the PCH framework, to formally prove that no sensitive information is leaked through the primary output or the Trojan side channels. The development of a full security property library is still an active topic under investigation.

Besides the RTL code trust validation, the proof-carrying based information assurance schemes were later adapted to support gate level circuit netlists, an effort largely expanding the application scope of proof-carrying hardware in IP protection [68]. Leveraging the new gate-level framework, the authors in [68] formally analyzed the security of design-for-test (DFT) scan chains, the industrial standard testing methods for fabricated chips and have formally proven that a circuit with scan chain inserted can violate data secrecy property. Although security concerns caused by DFT scan chains have been under investigation for decades with various attacking and defense methods being developed [69–77], it is the first time that the vulnerability of scan chain inserted designs has been formally proven (Note that RTL verification methods can rarely touch scan chains because scan chains are inserted at the netlist). The same framework is also applied to built-in-self-test (BIST) structure to prove that BIST structure can also leak internal sensitive information.

3.2. SAT Solver Based Verification Methods

Besides the proof-carrying methods, the SAT solver is also used in RT-level code security verification. For example, a four-step procedure to filter and locate suspicious logic in third-party IPs was proposed in [53]. In the first step, easy-to-detect signals were removed using functional vectors generated by a
sequential Automatic Test Pattern Generation (ATPG). In the next step, hard-to-excite and/or propagate signals were identified using a full-scan N-detect ATPG. To narrow down the list of suspected signals and identify the actual gates associated with the Trojan, a SAT solver was used in the third step for equivalence checking of the suspicious netlist containing the rarely triggered signals against the netlist of the circuit exhibiting correct behavior. At the final step, clusters of untestable gates in the circuit were determined using the region isolation approach on the suspected signals list. Another multi-stage approach, which included assertion-based verification, code coverage analysis, redundant circuit removal, equivalence analysis and use of sequential ATPG was proposed in [78] for suspicious signal identification. This approach was demonstrated on a RS232 circuit and the efficiency of the approach in detecting Trojan signals ranged between 67.7% and 100%.

4. Counterfeiting Prevention and IC Protection

Besides hardware Trojans, the IC supply chain also suffers from security threats at various points such as IP piracy, IC cloning, hardware backdoors, and counterfeit chips. Many techniques have been proposed to deter such attacks, e.g., [3–7,14–19,22,25–27,38,47–51,79]. Further, realizing the severity of counterfeit chips in critical systems, hardware security researchers started investigating into this area, hoping to detect faked or illegally marked chips before their deployment. Data analysis and machine learning methods have been used to recover chip identification [80,81]. On-chip aging sensors are a popular solution for counterfeit chip isolation [82]. Another emerging problem is IC overproduction and IP piracy [83–85], which occurs primarily because of lack of oversight and/or direct involvement in fabrication after passing the design over to the foundry. Effectively, there are no feasible solutions to determine if the foundry is producing exactly what the consumer ordered, or if they have over-produced the chips. To solve this problem the idea of split manufacturing was proposed [86,87], which enabled IP vendors to rely on overseas manufacturing services and not send the entire design information. This concept is recently expanded from digital domain to RF/analog circuits [88]. In this method, front-end of the line (FEOL) which requires advanced technology is fabricated overseas but the back-end of the line (BEOL) is added in domestic foundries such that the overseas foundries only learn partial design data. However, the effectiveness of this method is still under discussion [89,90].

As the design sector has become one of the most profitable stages of the IC design flow, and with the improvement of reverse engineering toolsets [91], protection of circuit designs and IP cores are receiving increasing attention in IC supply chain security. Indeed, IP piracy has become a serious threat to the IC industry. Toolsets that enable reverse engineering of circuits and IP core duplication are becoming more prevalent [91]. Various protection methods have been developed to prevent these kind of attacks. One IP protection method is Camouflaging [92–94] which relies on layout-level obfuscation that makes the recovery of a circuit’s structure through reverse engineering much more challenging [95]. Unfortunately, the overhead of CMOS camouflaging gates is often significant—especially as the level of protection increases. (A XOR + NAND + NOR camouflaging gate has 5.1X–5.5X power, 1.1X–1.6X delay, and 4X area compared to a conventional NAND or NOR gate [95].) Design obfuscation and logic encryption are other solutions that could prevent attackers from easily recovering/reproducing circuit designs without obfuscation keys [84,85,96,97]. While this method has proven to be robust to attacks.
(IP piracy could only occur if attackers know both the netlist and the keys) performance overhead and layout re-designing are challenges. For example, a fault analysis-based logic encryption method can increase power consumption by up to 188%, delay by up to 284%, and area by as much as 242% (assuming ISCAS testbenches) [85].

Side-channel analysis and fault injection is another major threat to hardware security, especially for circuits containing sensitive information. Without physical intrusion, attackers can recover internal signals leveraging static/differential analyses on side channels such as timing [98,99], power consumption [100], and electromagnetic emissions [101,102]. Besides passive side-channel analysis, cryptographic circuits are also vulnerable to power supply-based fault injections [103]. In order to counter these attacks, researchers have developed various logic circuitry and on-chip sensors to balance the side-channel signals and to detect signal anomalies [104–107]. However, even with the help of design optimization methods, the existing countermeasures still incur a high performance overhead [108].

5. Physical-Unclonable Functions

Orthogonal to these hardware security areas is the development of security primitives, which are investigated due to their high efficiency and low cost compared to software solutions.

The leading example is the physically unclonable function (PUF) which leverages device mismatches caused by process variations to generate unique identities, often Challenge-response pairs, for each chip [109]. While there are many criteria toward evaluating PUF designs, the main criteria which are closely related to hardware security applications are listed below:

- **Randomness.** The randomness is the measurement indicating how random the responses are given any input patterns.
- **Uniqueness.** The uniqueness is another critical criterion indicating how robust the PUF design is under the different environmental conditions and/or noises.
- **Enhanced security.** The security property measures how resilient the PUF designs are under attack. In fact, various attack methods, including machine learning and device modeling, have been developed recently to break PUF designs and predict PUF responses [110–112].

A large amount of work has been proposed recently to improve these metrics using error correction algorithms [113] and non-MOSFET technologies [114,115]. Since the design and evaluation of PUFs have been widely covered in other literature, we refer to [116–119] for readers who are interested in PUF development and implementations.

6. Emerging Devices in Hardware Security

While existing hardware security methods are mostly in the areas of chip manufacturing, circuit design, and circuit testing, future trends will cover a broader area in order to grant hardware active roles in system level protection. Specifically, two hardware security research areas will be introduced to show hardware security from a single layer to cross layers, emerging transistors in hardware security applications and hardware-assisted cybersecurity solutions, including emerging devices in hardware security applications and hardware-assisted cybersecurity.
Researchers in emerging devices are currently investigating their applications in broader security areas. Due to the availability of a large number of emerging device models such as graphene transistors, atomic switches, memristors, MOTT FET, spin FET, nanomagnetic and all-spin logic, spin wave devices, OST-RAM, magnetoresistive random-access memory (MRAM), spintronic devices, etc. [120], two fundamental questions have recently been raised related to their applications in the hardware security domain: (1) Can emerging technology provide a more efficient hardware infrastructure than CMOS technology in countering hardware Trojans and IP piracy? (2) What properties should the emerging technology-based hardware infrastructure provide so that software level protection schemes can be better supported? Most work in emerging technologies for security purposes to date has been in PUFs [115] and cryptographic primitives. However, PUFs essentially leverage device-to-device process variation. In some sense this suggests that greatly mismatched devices are more useful. Orthogonal to these efforts, researchers try to leverage the unique properties of emerging technologies, other then relying on mismatched devices, for IP protection and hardware attack prevention [121].

7. Hardware-Assisted Computer Security

In addition to the circuit-level IC supply chain protection methods, hardware platforms can also be involved in software-level protection schemes. Cybersecurity researchers often rely on layered security protection methods and have developed various methods to protect a higher abstract layer (e.g., guest OS) through security enhancements at a lower abstract layer, (e.g., virtual machine monitor or hypervisor) [122–124]. Through this chain, cybersecurity protection schemes have been pushed downward from guest OS to hypervisors. Following this trend, new methods are under development through which the hardware infrastructure is modified to directly support sophisticated security policies such that protection schemes operating at the system-level will be more efficient [125,126]. In fact, security-enhanced hardware supporting cybersecurity protections have become quite popular in both academic research and industrial products recently [28,29,127].

7.1. ARM TrustZone

The TrustZone approach to enabling trusted computing is based on a trusted platform, wherein a hardware architecture supports and enforces a security infrastructure throughout the system. Instead of protecting assets in a dedicated hardware block, the TrustZone architecture enforces system-wide security extended to any component of the system. In effect, the TrustZone architecture aims to provide a security framework to enable the construction of a programmable environment that allows the confidentiality and integrity of application assets from a range of attacks [28].

The TrustZone implementation relies on partitioning the SoC’s hardware and software resources so that they exist in two worlds: secure and non-secure. Hardware supports access control and permissions for the handling of secure/non-secure applications and the interaction and communication among them. The software supports secure system calls and interrupts for secure run-time execution in a multitasking environment. These two aspects ensure that no secure world resources can be accessed by the normal world components, except through secure channels, enabling an effective wall-of-security
to be built between the two domains. This protection extends to I/O connected to the system bus via the TrustZone-enabled AMBA3 AXI bus fabric, which also manages memory compartmentalization.

7.2. Intel SGX

Intel SGX (software guard extension) describes extensions added to the Intel architecture in order to enforce memory access policies and permissions. These changes enable applications to execute with confidentiality and integrity in the native operating system (OS) environment. Userspace compartmentalization is accomplished via instruction set architecture (ISA) extensions for generating hardware-enforceable containers at a granularity determined by the developer, either fine or coarse. These extensions allow an application to instantiate a protected container known as an enclave, which defines a protected area in the applications address space. These containers, while opaque to the OS, are still managed by the OS. Attempted accesses to the enclave memory area from software not resident in the enclave are prevented. This is true for privileged software such as virtual machine monitors (VMMs), the BIOS, and OS. The SGX extensions offer improved architectural support in enclave memory access semantics and address mapping protection [29,128].

As mentioned, an enclave is a protected area in the application’s address space that provides confidentiality and integrity even in the presence of privileged software. Before the enclave is built, the enclave code and data is free for inspection/analysis. Once the protected portion is loaded into an enclave its code/data is measured and is protected against all external software access. An application can prove its identity to a remote party and can be securely protected with keys and credentials, as well as enclave and platform specific keys.

7.3. CHERI

The CHERI (capability hardware enhanced RISC instruction) extension attempts to provide fine-grained compartmentalization of memory in hardware while maintaining software compatibility. Previous approaches providing fine-grained compartmentalization via software virtualization result in significant performance overhead. Reducing the overhead through hardware compartmentalization in the past has only provided coarse-grained protection. Memory management unit (MMU) page protection or the stack frame are two examples of compartmentalization that strike a balance between improved performance while still offering virtualized protection, but at the expense of granularity [129].

The CHERI project extends the 64-bit MIPS ISA (instruction set architecture) with byte granularity memory protection through a hybrid, capability-based addressing scheme with MMU virtual memory support. LLVM compiler and FreeBSD OS modifications have been implemented to support the CHERI extension. Of primary concern to the CHERI project is memory protection via eight requirements: un-privileged use, fine granularity, un-forgeability, access control, segment/domain scalability, incremental deployment, and pointer safety. Enforcing protection as the common case with infrequent system calls refers to un-privileged use. Fine granularity should accommodate data that is densely packed and should be capable of handling odd numbers of bytes/words. Software is unforgeable in that it is unable to increase its permissions. The hardware should monitor and enforce protected memory regions and permission to satisfy access control. Per segment and domain, the memory storage
overhead should scale gracefully with the number of protected regions and the frequency of use respectively. Finally, extant user-space software should run without recompilation, even if protected components such as the shared library make use of fine-grained protection to be incrementally deployed. Pointer safety is enforced by associating protection properties with object references similar to fat pointers. Every pointer encodes the address to which it points and the base and bounds of the frame to which it points.

7.4. LowRISC

LowRISC aims to be a fully open-source SoC based on the 64-bit RISC-V instruction set architecture \[130,131\]. The primary design goals include targeting a 40/28 nm technology node, being released under an open license, novel security features, programmable I/O, AMBA bus, and Linux support. Additionally, the SoC will be manufactured in volume as a low-cost development board. Primarily motivated by the open-source community and innovative startups, the Cambridge team hopes to offer a research platform supported by both academics, professionals and hobbyists.

Albeit at the early design phases, the lowRISC developers have released several details about the proposed SoC that they believe will achieve both security and high performance: tagged memory and minion cores. Note, these are planned, unimplemented design targets so that evaluation and performance metrics have not yet been presented. Effectively, “minion” cores are simple processors with predictable features and direct access to I/O that are intended to be used to filter I/O and run tasks with limited performance requirements. Tagged memory associates metadata with each memory location and is to be used to enforce fine-grained memory access restrictions.

In the lowRISC tagged memory implementation, every memory address is appended with tag bits. In this way, the lowRISC implementation supports basic process isolation. To protect against memory corruption—of the return address, vtable pointers, and code pointers on the stack/heap—the compiler is modified to generate instructions to mark vulnerable locations. This may take the form of read-only protection in the case of the return address or vtable pointers, or locking/unlocking code pointers to protect use-after-free vulnerabilities. Additionally, control-flow integrity checks can be applied to indicate valid targets based on the program’s control flow graph. The minimum requirement for the broad security policy enforcement offered by lowRISC includes ISA extension, modification to the compiler, memory allocator, a libc environment, and updating the kernel virtual memory system to handle the tag bits.

8. Conclusions

In this survey paper, the history of the hardware security research had been introduced in detail. The current research efforts were also elaborated on as well as the future trends in this emerging area. This summary provides a reference for hardware security research and, hopefully, serves as a clear guide for researchers to join this area and push the boundary further. Through this paper, we hope to attract more researchers to join this area and develop more innovative and fundamental solutions to solve hardware level threats and, as the final goal, to ensure the trustworthiness of the “root-of-trust”.
Acknowledgments

This work was supported in part by the National Science Foundation grant (CNS-1319105).

Conflicts of Interest

The authors declare no conflict of interest.

References


131. Asanović, K.; Patterson, D.A. Instruction Sets Should Be Free: The Case For RISC-V; Technical Report for No. UCB/EECS-2014-146; EECS Department, University of California: Berkeley, CA, USA, 6 August 2014.

© 2015 by the author; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).