Invited: Protecting the Supply Chain for Automotives and IoTs

Sandip Ray
University of Florida
Gainesville, FL 32611. USA
sandip@ece.ufl.edu

Wen Chen
NXP Semiconductors Inc.
Austin, TX 78735. USA
wen.chen@nxp.com

Rosario Cammarota
Qualcomm Technologies Inc.
San Diego, CA 92121. USA
ro.c@qti.qualcomm.com

ABSTRACT

Modern automotive systems and IoT devices are designed through a highly complex, globalized, and potentially untrustworthy supply chain. Each player in this supply chain may (1) introduce sensitive information and data (collectively termed "assets") that must be protected from other players in the supply chain, and (2) have controlled access to assets introduced by other players. Furthermore, some players in the supply chain may be malicious. It is imperative to protect the device and any sensitive assets in it from being compromised or unknowingly disclosed by such entities. A key — and sometimes overlooked — component of security architecture of modern electronic systems entails managing security in the face of supply chain challenges. In this paper we discuss some security challenges in automotive and IoT systems arising from supply chain complexity, and the state of the practice in this area.

Keywords

Secure keys, Hardware Trojans, Design for Test, Design for Debug, Key Provisioning

1. INTRODUCTION

We are living in a world pervaded by smart, connected electronic systems. The so-called era of Internet-of-Things (IoT) — defined as the time when the number of electronic devices connected to the Internet exceeds the human population — started around 2008 [3]. A decade into it, the number of such devices has been growing at a rate faster than any sector at any point in the human population, with estimates ranging from 50 billion to 100 billion devices by 2020, and going to trillions within another decade. Connected devices in the IoT era range from the big (*e.g.*, self-driving cars, connected convoys, etc.), to the small (*e.g.*, light bulbs, baby monitors, wearables, etc.) and to the really tiny (*e.g.*, miniaturized devices with attached sensors that perform as "smart dust"). Furthermore, we continue to develop applications where these devices continually monitor, collect, aggregate, analyze, and communicate some of our most sensitive, personal information (*e.g.*, sleep patterns, health information, browsing history, locations, contacts). Security of these devices is consequently of paramount importance. It is critical to ensure that the sensitive information stored and communicated by these devices is not vulnerable to unauthorized, malicious access [12].

DAC '18 June 24–29, 2018, San Francisco, CA, USA

ISBN 978-1-4503-5700-5/18/06…$15.00

DOI: https://doi.org/10.1145/3195970.3199851

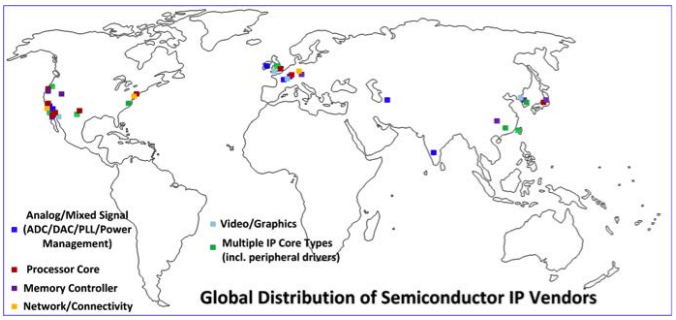

Figure 1: A SoC containing IPs from entities distributed across the globe

A key component of security of modern computing devices entails handling the complex supply chain involved in their development and production. Most electronic devices today are developed using the System-on-Chip (SoC) design paradigm where a system is created by connecting and integrating a collection of pre-designed hardware and hardware/software blocks, often referred to as Intellectual Property cores or "IP cores" or simply "IPs". Such IPs are procured today from a diversity of vendors distributed across the globe (refer to Fig. 1). In particular, the global market for third-party semiconductor IPs reached more than 2.1 billion in late 2012, with trends towards even sharper gradient thereafter [9]. Furthermore, the integration of the SoC design, physical design, testing, validation, fabrication, and production are each globally distributed. This raises intriguing questions related to security. For instance, there are subtle and strong restrictions on how assets introduced by one player in the supply chain can be accessed by downstream participants. Addressing these constraints is a critical aspect of security architecture of modern electronic systems.

In this paper, we discuss the range of problems arising from trust and security requirements from a complex supply chain in today's automotive and IoT systems. Our goal is not to be comprehensive, but to provide a flavor of chal-

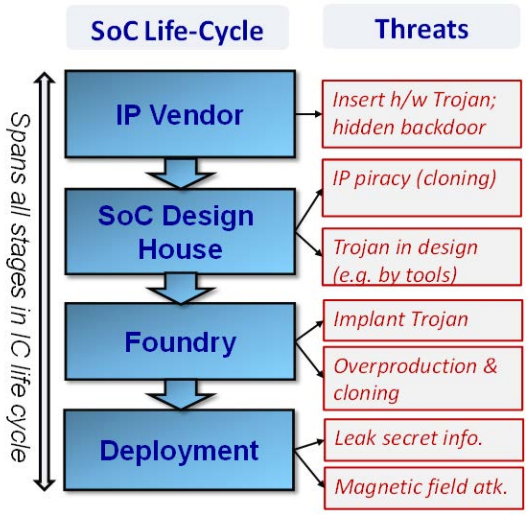

Figure 2: Various Players in SoC Design Supply Chain, and Security Threats Introduced by Each Player [10].

lenges involved, approaches employed in current state of the practice to address these challenges, and the limitations and deficiencies of these approaches. We hope that such a consolidated treatment will help unify several disparate research threads in this highly complex area, and spur further innovations.

The remainder of the paper is organized as follows. Section 2 discusses some of the traditional and well-explored supply chain challenges. In Section 3 we consider the problem of protecting assets from individual players in the supply chain, and the challenges involved. We discuss the additional complexities arising from test and debug requirements in Section 4. We discuss some of the current industrial approaches to combat these issues in Section 5. We conclude in Section 6.

2. SUPPLY CHAIN AND TRADITIONAL SECURITY CHALLENGES

Supply chain security has been an active topic of research in the hardware security community, with several excellent treatises [13, 1, 7, 2, 4]. The security threats considered in the literature include Trojan insertions, IP piracy, cloning, counterfeit ICs, and overproduction, among others. In this section, we summarize some of the key security issues.

Fig. 2 illustrates the various players in the SoC design and fabrication supply chain as well as the potential security threats introduced by different players. These threats are aggravated with the rapid globalization of the SoC design, fabrication, validation and distribution steps. Given the growing complexity of the IPs, as well as the SoC integration process, SoC integration designers increasingly tend to treat IPs as black boxes and rely on the IP vendors on the structural/functional integrity of these IPs. However, such design practices greatly increase the number of untrusted components in an SoC design.

Hardware IPs acquired from untrusted third-party vendors can have diverse security and integrity issues. An adversary inside an IP design house (*e.g.*, a rogue designer) involved in the IP design process can insert a malicious implant or design modification to incorporate hidden or undesired functionality. These Trojans can act as a backdoor, as a covert channel, or compromise the functional/parametric properties of an SoC in various ways. In addition, since many of the IP providers are small vendors working under highly aggressive schedules, it is difficult to enforce a stringent IP validation requirement in this eco-system. Computer Aided Design (CAD) tools pose similar trust issues to the SoC designers. Such tools are designed to optimize a design for performance, power, and area. These optimizations can introduce new vulnerabilities [8].

Even in the absence of untrusted vendors, many IPs suffer from lack of "lineage traceability", *i.e.*, a way for the SoC integration house to trace how (and by whom) the IP was developed and how much trust can be put on its secure functionality. For example, the lineage of an IP may be lost because of mergers and acquisitions of its original vendors, or because it has been procured from the open source potentially with long-forgotten development roots. Consequently, it is a challenge to ensure such IPs do not introduce vulnerabilities or even estimate the risk involved in integrating them to a target SoC.

Furthermore, modern SoC designs have many (non-functional) design features that may introduce vulnerabilities, *e.g.*, intentional information leakage through hidden test/debug interfaces, hidden in partially specified functionalities [4], or side-channels through power/performance profiles [6]. Rogue designers in an untrusted design facility, *e.g.*, in case of a design outsourced to an untrusted facility for Design-for-Test (DFT) or Design-for-Debug (DFD) insertion, can compromise the integrity of an SoC design through insertion of stealthy hardware Trojan.

Finally, many SoC manufacturers today are fabless and hence must rely upon external untrusted foundries for fabrication service. An untrusted foundry has access to the entire design and thus brings in several serious security concerns, which include reverse engineering and piracy of the entire SoC design or the IP blocks as well as tampering in the form of malicious alterations or Trojan attacks. During distribution of fabricated SoC designs through a typically long supply chain, consisting of multiple tiers of distributors, wholesalers, and retailers, the threat of counterfeits is a growing one. These counterfeits can be low-quality clones, overproduced chips in untrusted foundry, or recycled ones. Even after deployment, the systems are vulnerable to physical attacks, *e.g.*, side-channel attacks which target information leakage, and magnetic field attacks that aim at corrupting memory content to cause denial-of-service attacks.

3. PROTECTING ASSETS ACROSS SUPPLY CHAIN

The threats discussed in Section 2 pertain to a malicious player in the supply chain potentially introducing vulnerability to enable leakage or corruption of assets (*e.g.*, through Trojans), or disrupting the production process itself (*e.g.*, through counterfeit, cloning, overproduction, etc.). Another aspect of the challenges introduced by a complex supply chain is that the assets introduced by any individual player in the supply chain must be protected from other players that subsequently encounter the SoC. In this section, we discuss that challenge in some detail.

Consider an electronic part developed for an automotive system. Typically, such a part would be developed by some electronic part vendor, and would subsequently go through several "tiers" of part suppliers, eventually to an automotive manufacturer who would integrate the part into an automobile that eventually reaches the customer. There may be other players in the supply supply chain, including various OEMs (Original Equipment Manufacturers), ISPs (Independent Software Vendors), etc. Each player in this process can introduce several sensitive assets to the part. These include cryptographic keys, Digital Rights Management (DRM) keys (for infotainment parts), proprietary firmware, and software, various in-field debug techniques, etc. [10]. Consequently, these assets should be protected, not only after the system is in-field (*i.e.*, with the customer), but also when it is with the subsequent players in the supply chain. In particular, the following constraints should be considered:

1. Assets introduced by the vendor should not be accessible to suppliers, automotive manufacturer, or end user.
2. Assets introduced by a supplier or automotive manufacturer should not be accessible to any other party, including the original vendor of the part.
3. All assets should be protected against side-channel attacks (*e.g.*, voltage, temperature, or clock glitch attacks).
4. Customer and third-party software should be protected against unauthorized access.

To emphasize the subtlety involved in ensuring these constraints, it may be worthwhile to examine the constraint 2 above a bit more closely. Why is it hard to ensure that assets introduced by OEMs or automotive manufacturers are protected from the original supplier of the part? After all, the part goes *from the vendor* to the supplier and subsequently to the automotive manufacturer, the supplier does not see the secrets introduced by these subsequent players, *is that right?*

Well, it is wrong! One potential scenario where the supplier in fact can see the assets introduced by the subsequent players in the supply chain entails field return. If the part is returned from field, — possibly in response to a problem found after deployment — it can potentially include assets of every player in the supply chain. Thus, *it is challenging to ensure that the supplier would not have access to any of the assets by other players while still being able to debug the problems that caused the return.*

There are two other factors that potentially exacerbate the problem. First, not all the assets are introduced in hardware: a significant portion of the assets is in fact introduced through firmware. Second, assets are not all static. While some are provisioned by the various stake-holders (*e.g.*, in fuse controllers, e-wallet, etc.), others are created as the system executes. For example, a master key may be provisioned statically and other cryptographic keys derived from them during the boot process. An upshot of this is that protection of assets must be extended to activities like firmware load and system boot, and these must account for whether the system is with the supplier (at first fabrication or after field return), OEM, manufacturer, or in-field.

How is all this achieved today? The lifetime of the system is divided into a number of stages (often referred to ads "life-cycle stages"), each corresponding to the time when a specific player in the supply chain has access to the part. Consequently, each stage defines protection requirements for various assets, which are typically implemented through programmable fuses or flash memories. Unfortunately, there is no systematic way today to do this programming. It remains up to the creativity of the security architect to develop mechanisms that enforce various subtle invariants that ensure assets are protected from unauthorized access (for that life cycle). Given the complexity of today's SoC designs and protection requirements, it is unsurprising that there may be bugs in these mechanisms.

4. TEST AND DEBUG CHALLENGES

There is one crucial factor that makes asset protection particularly complex along the supply chain: the need for various participants to debug and test the part. The conflicts and trade-offs between security and debug are well-known and have been considered elsewhere [11]. Here we only focus on the impact of the trade-off on the supply chain.

Virtually all modern SoC designs contain an interface to enable observability and controllability of internal signals for post-silicon and in-field debug. One standard debug and test interface that is available to virtually all parts is the JTAG interface [5], but other interfaces are also available. These interfaces provide the user structured access to internal architectural and design features (*e.g.*, scan chain, various design-for-debug features, etc.) for the purposes of functional verification, manufacturing test, and related activities. Since these activities entail observability of internal states of the design (and consequently of assets stored) there are strong restrictions on how each player in the supply chain can access these interfaces, *e.g.*, the original part vendor may have unrestricted authorization to access these interfaces while a supplier might only access them with a password provided by the part vendor (and is therefore dependent on vendor's authorization for access). The level of access of each of these interfaces for each participant of the supply chain must be defined by accounting for the assets in the the system when that participant has access to the part, the access restrictions on those assets, the kind of access needed for the respective participant to effectively perform their role (*e.g.*, do effective test and debug). For example, an OEM that introduces custom software on a part must have the ability to debug that software while not having access to the supplier keys.

To add a final twist to the complexity involved, recall from above that assets are protected by fuse or flash programming at each life-cycle stage. When a part goes from one participant of the supply chain to the next, the corresponding fuses are programmed to provide requisite protection to the assets. The problem is that this fuse/flash programming is actually performed through the debug interface, *e.g.*, by issuing a sequence of JTAG commands. Since the programming also updates the life-cycle stage, — and thereby constrains how the debug interface itself can be used at the end of the programming — the use of the debug interface to perform this programming often introduces a complex cyclical dependency between the access requirements for programming and the access restrictions for the next stage in the life-cycle.

5. EMERGENT DIRECTIONS

As should be clear from the preceding sections, protecting assets as an electronic system goes through the various participants in the supply chain is a subtle and complex enterprise. Addressing this problem requires a cooperation between both *security architecture* and *validation*. In this section, we briefly recount some approaches in both directions, as well as the challenges encountered.

Architectural Isolation: Trust Provisioning: Trust provisioning is the idea in which assets are provisioned by various stake-holders through a specific, centralized trust model. The trust model is typically defined by the supplier of the part who is also responsible for the architecture that enables various stake-holders to insert assets at different life-cycle stages. The provisioning mechanism guarantees that (1) a service that does not need an asset does not get access to it, and (2) access and update to each asset satisfies the trust model.

Trust provisioning has emerged as a promising vision towards developing asset protection policies in emerging automotive and IoT systems that involve a complex globalized supply chain. However, the approach is still in infancy, and further research is awaited to develop a robust, provable architecture for trust provisioning.

Validating Life-Cycle Isolation: Practice and Challenges: A key aspect of security validation for modern SoC designs is the so-called "life-cycle isolation", *i.e.*, checking that an asset is not accessed in a life cycle stage in which its access is not authorized. In principle, this is no different from traditional access control and consequently is within the purview of modern security validation tools. However, performing this in practice on a (pre-silicon) SoC design involves a number of complexities. For example, a key component of access restrictions involves test and debug interfaces which are not available in a design at (say) the RTL level, making it difficult to apply formal tools that typically work at that level. Second, note that it is possible to move from one life cycle to another, so one validation target is to ensure that this transition does not happen without the proper fuse programming. However, since this transition may involve multiple reset sequences, it is difficult to perform this check with current tools. Finally, note that the fuse programming involved in transitioning a part from one life cycle stage to the next involves an authentication process. This authentication is typically performed in software, which is difficult to verify in pre-silicon simulation.

Note that the above discussions only scratch the surface of the complexity involved in developing architectural and validation technologies for protecting systems in the presence of supply chain challenges; they are not meant to be comprehensive. However, perhaps they do provide a flavor of the nature and spectrum of challenges involved, and the kind of concerns to which attention must be paid to develop a comprehensive solution to these challenges.

6. CONCLUSION

We have discussed some of the challenges arising from the complex supply chain involved in the development of modern automotive and IoT devices. In addition to traditional hardware security issues (*e.g.*, Trojans, cloning, counterfeit), one must also account for protecting various sensitive information introduced by various players in the supply chain. Addressing this is a complex, subtle, and vexing problem. A comprehensive solution will require a re-thinking of the architecture, and comprehending and accounting for the trade-offs necessary among various stakeholders' interests.

7. REFERENCES

[1] S. Bhunia, S. Ray, and S. Sur-Kolay. *Fundamentals of IP and SoC Security: Design, Validation, and Debug*. Springer, 2017.

[2] S. Bhunia and M. Tehranipoor. *The Hardware Trojan War: Attacks, Myths, and Defenses*. Springer, 2017.

[3] D. Evans. The internet of things - how the next evolution of the internet is changing everything. *White Paper. Cisco Internet Business Solutions Group (IBSG)*, 2011.

[4] N. Fern, I. San, Ç. K. Koç, and K. Cheng. Hiding hardware trojan communication channels in partially specified soc bus functionality. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 36(9):1435–1444, 2017.

[5] IEEE Joint Test Action Group. IEEE Standard Test Access Port and Boundary Scan Architecture. *IEEE Std.*, 1149(1), 2001.

[6] E. Messmer. RSA security attack demo deep-fries Apple Mac components, 2014. See URL http://www.networkworld.com/news/2014/022614-rsa-apple-attack-279212.html.

[7] P. Mishra, S. Bhunia, and M. Tehranipoor. *Hardware IP Security and Trust*. Springer, 2017.

[8] A. Nahiyan, K. Xiao, D. Forte, Y. Jin, , and M. Tehranipoor. AVFSM: A Framework for Identifying and Mitigating Vulnerabilities in FSMs. In *Design Automation Conference (DAC)*, 2016.

[9] G. Ramamoorthy. Market Share Analysis: Semiconductor Design Intellectual Property, Worldwide, 2012. See URL https://www.gartner.com/doc/2403015/market-share-analysis-semiconductor-design.

[10] S. Ray, E. Peeters, M. Tehranipoor, and S. Bhunia. System-on-Chip Platform Security Assurance: Architecture and Validation. *Proceedings of the IEEE*, 106(1):21–37, 2018.

[11] S. Ray, J. Yang, A. Basak, and S. Bhunia. Correctness and Security at Odds: Post-silicon Validation of Modern SoC Designs. In *Proceedings of the* 52*nd Annual Design Automation Conference*, 2015.

[12] SIA. Semiconductor Research Opportunities: An Industry Vision and Guide. Technical report, Semiconductor Industry Association, 2017.

[13] M. Tehranipoor and C. Wang. *Introduction to Hardware Security and Trust*. Springer, 2011.