Towards Secure Composition of Integrated Circuits and Electronic Systems: On the Role of EDA

Johann Knechtel[1], Elif Bilge Kavun[2], Francesco Regazzoni[3], Annelie Heuser[4], Anupam Chattopadhyay[5], Debdeep Mukhopadhyay[6], Soumyajit Dey[7], Yunsi Fei[8], Yaacov Belenky[9], Itamar Levi[10], Tim Güneysu[11], Patrick Schaumont[12], and Ilia Polian[13]

[1]johann@nyu.edu – NYU Abu Dhabi, UAE; [2]e.kavun@sheffield.ac.uk – University of Sheffield, UK;
[3]regazzoni@alari.ch – ALaRI, University of Lugano, Switzerland;
[4]annelie.heuser@irisa.fr – Univ Rennes, Inria, CNRS, IRISA, France;
[5]anupam@ntu.edu.sg – NTU, Singapore; [6,7]debdeep@cse.iitkgp.ac.in, soumya@cse.iitkgp.ac.in – IIT Kharagpur, India;
[8]yfei@ece.neu.edu – NEU, Boston, USA; [9]yaacov.belenky@intel.com – Intel, Israel;
[10]itamar.levi@biu.ac.il – BIU, Ramat Gan, Israel; [11]tim.gueneysu@rub.de – RUB & DFKI, Bochum & Bremen, Germany;
[12]schaum@vt.edu – VT, Blacksburg, USA; [13]ilia.polian@informatik.uni-stuttgart.de – University of Stuttgart, Germany

***Abstract*—Modern electronic systems become evermore complex, yet remain modular, with integrated circuits (ICs) acting as versatile hardware components at their heart. Electronic design automation (EDA) for ICs has focused traditionally on power, performance, and area. However, given the rise of hardware-centric security threats, we believe that EDA must also adopt related notions like *secure by design* and *secure composition of hardware*. Despite various promising studies, we argue that some aspects still require more efforts, for example: effective means for compilation of assumptions and constraints for security schemes, all the way from the system level down to the "bare metal"; modeling, evaluation, and consideration of security-relevant metrics; or automated and holistic synthesis of various countermeasures, without inducing negative cross-effects.**

In this paper, we first introduce hardware security for the EDA community. Next we review prior (academic) art for EDA-driven security evaluation and implementation of countermeasures. We then discuss strategies and challenges for advancing research and development toward secure composition of circuits and systems.

I. Introduction

Electronic systems are at the heart of our modern societies which are heavily reliant on ubiquitous information technology (IT). Nowadays, however, an alarmingly large number of security risks are associated with electronic systems. Ensuring confidentiality, integrity, and availability—the three key pillars for IT security—directly within the hardware of electronic systems represents a wide-ranging task that is crucial, yet quite challenging. The related field of hardware security has been driven traditionally by the cryptography community, and rightfully so; the formal security promises of any cryptographic algorithm may fail relatively easily once the physical realities of hardware come into play. For example, it is well known that cryptographic algorithms leak sensitive information when subjected to side-channel attacks [1] or fault-injection attacks [2]. While at least parts of the electronic design automation (EDA) community have become aware of these and other threats over the years, and also proposed some EDA measures to counter them, we argue that more concerted efforts are required.

TABLE I
Security Threats for ICs and Related Roles of EDA

Threat Vector	Time of Attack	Role of EDA
Side-channel attacks	Runtime	Evaluation, mitigation at design time
Fault-injection attacks	Runtime	Evaluation, mitigation at design time
Piracy of design intellectual property (IP); counterfeiting of ICs	Manufacturing; in the field	Mitigation at design time
Hardware Trojans	Design; manufacturing	Mitigation, verification at design time; preparing for testing, inspection

In this paper, we aim to educate the broader EDA community on the different security threats arising for integrated circuits (ICs) throughout their life cycle, i.e., during design, manufacturing, and at runtime. In Table I, we list the threats covered in this paper and the roles we see for EDA in general.

We motivate in Sec. II, we review the prior art in some detail in Sec. III, and we discuss strategies and challenges for advancements in Sec. IV. Overall, we call for paradigms like *secure by design* and *secure composition of hardware*, i.e., for efforts to account holistically for security notions along with traditional notions of design optimization.

II. Background and Motivation

A. Security Threats and Overview on Countermeasures

Next, we introduce briefly the security aspects we cover in this paper. This section is an overview and not comprehensive; we discuss related prior art in Sec. III in more detail.

1) Side-Channel Attacks (SCAs): SCAs exploit information leakage from measurable physical channels and sensitivities of (i) the circuitry itself or (ii) the architecture. For example, concerning (i), advanced encryption standard (AES) implementations are well-known to be vulnerable to power SCAs when unprotected [1]; concerning (ii), modern microprocessors leak

978-3-9819263-4-7/DATE20/©2020 EDAA

information through timing behaviour of caches, also related to speculative execution [3].

Most countermeasures apply some kind of "hiding" or *masking*, i.e., diffusion of the information leakage, by various means taken across different levels, starting from the system level and ranging down to gates/registers [4]. Formal approaches to masking, e.g., [5], refer to splitting the computation variables into sections or *shares* such that internal computations are never performed jointly on all shares.

2) Fault-Injection Attacks (FIAs): FIAs induce faults to aid in deducing sensitive information. This includes direct, invasive fault injection, e.g., by laser light [6] or electromagnetic waves [7], as well as indirect, architectural fault injection, e.g,. by repetitive writing to particular memory locations [8].

Countermeasures can be separated into detection of FIAs at runtime versus FIA mitigation at design time (e.g., [9], [10]).

3) Piracy of Design IP, Counterfeiting of ICs: Such attacks related to outsourced IC supply chains can be carried out by various adversaries, ranging from designers, foundry or test facility employees, and even to end-users.

Popular countermeasures against IP piracy are logic locking, split manufacturing, and camouflaging [11]. Both split manufacturing and camouflaging alter the manufacturing process to protect against untrusted foundries and malicious end-users, respectively. In contrast, logic locking works at the design level to protect against untrusted foundries and malicious end-users (although the latter relies on tamper-proof memories, which can become targets themselves). Popular countermeasures against counterfeiting include watermarking and physically-unclonable functions (PUFs) [12].

4) Hardware Trojans: Given that IC supply chains are outsourced, adversaries at various entities could also introduce malicious hardware modifications, known as Trojans.[1] The notion of Trojans is wide-ranging [13]—it describes malicious modifications that are (i) working at the system level, register-transfer level (RTL), gate/transistor level, or the physical level; (ii) seeking to leak information, reduce the IC's performance, or disrupt the IC's working altogether; (iii) are always on, triggered internally, or triggered externally; etc.

Countermeasures can be classified into Trojan detection, conducted pre-silicon and/or post-silicon, and Trojan mitigation. The former relies on testing, verification, and inspection, whereas the latter includes security features to improve testability/inspection [13] or information-flow tracking [14], etc.

B. Classical EDA Flows and Security Fallacies

In Fig. 1, a classical EDA flow is shown in overview. Various EDA tools as well as design components and technology libraries are involved, which are all provided by potentially malicious third parties. This presents clearly one of the threats for secure composition of ICs. For example, Trojans could be introduced directly by adversarial designers, indirectly through untrustworthy third-party IP components, or even through "hacks" of EDA tools or the IT environment [13].

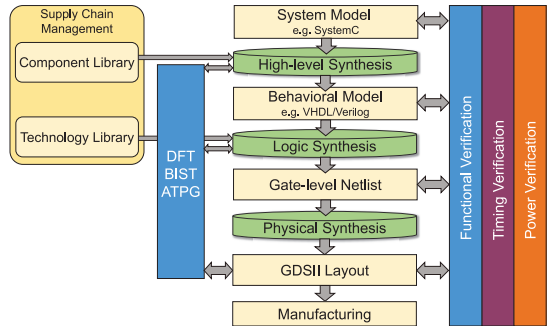

Fig. 1. Classical EDA flow, without security considered explicitly.

State-of-the-art EDA tools provide powerful solutions for simulation, verification, and testing, and they are also well-tailored to optimize any design for power, performance, and area (PPA). However, these tools are neither tailored yet to account for, e.g., information leakage exploited by SCAs, nor do they offer to incorporate countermeasures in a way that maintains optimization *and* security guarantees.

Motivational example: Here we show how classical EDA tools can undermine security. We consider the notion of *private circuits* [15] as an example, a scheme that guarantees confidentiality in the face of SCAs in a controllable and quantifiable manner. Without loss of generality, a bit a of sensitive information can be encoded as a vector (a_1, a_2, a_3), where $a = a_1 \oplus a_2 \oplus a_3$ and $\oplus$ denotes bitwise XOR. Any regular operations are implemented in encoded form and incorporate random bits $r_{i,j}$, where $1 \leq i, j \leq 3$. For example, the AND operation $c = a \wedge b$ on such vectors is computed as: $c_1 = a_1b_1 \oplus r_{1,2} \oplus r_{1,3}$ and $c_2 = a_2b_2 \oplus (r_{1,2} \oplus a_1b_2) \oplus a_2b_1 \oplus r_{2,3}$ and $c_3 = a_3b_3 \oplus (r_{1,3} \oplus a_1b_3) \oplus a_3b_1 \oplus (r_{2,3} \oplus a_2b_3) \oplus a_3b_2$.

The security promise by private circuits is based on the fact that all components of one such vector are never processed at the same time. Thus, an adversary cannot learn it from power measurements (or other side channels). Now, it is important to note that the order of computation, as indicated by parentheses, is critical for suppressing information leakage, even though it is irrelevant for correctness (as $\oplus$ is commutative). For the example of the AND operation, let us assume the synthesis tool implements c_3 such that the expression $a_3b_1 \oplus a_3b_2 \oplus a_3b_3 = a_3(b)$ is derived first and the random bits r_{ij} are added only later, then the computation will leak the value of b (Fig. 2). Regular, security-unaware tools may take such decisions easily, e.g., when it helps to improve timing.

Note that leakage can occur even when private circuits are synthesized in a security-aware manner, e.g., then due to delays and glitches for the random variables. An effective and well-known, yet limited, approach for leakage evaluation is *test vector leakage assessment (TVLA)* [16]; see Sec. III.

[1] Although it has been projected traditionally as the main threat scenario, the likelihood of Trojans being introduced at fabrication time is arguably very low. That is because any such endeavour, once detected, would fatally disrupt the reputation and business of the related foundry. Therefore, foundries can be expected to employ all organizational and technical means available to hinder unauthorized modifications by malign employees.

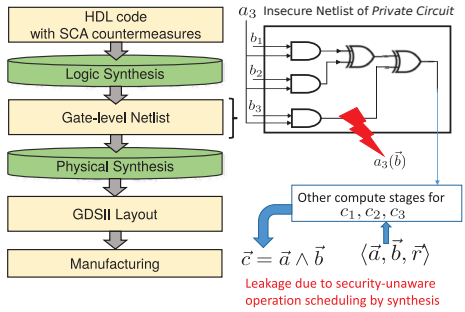

Fig. 2. Motivational example for the insecure nature of classical EDA tools.

C. Challenges and Tasks for Security-Centric EDA

As listed in Table I, we see potential for EDA tools to evaluate and mitigate various threats already at design time. Considering though the vastly different nature of these threats, it may seem impossible to provide comprehensive security-centric EDA flows. However, to make progress towards this ultimate goal, we argue that the EDA community should (continue to) focus on key challenges which are, among others:

- Evaluation and consideration of security-relevant metrics, with varying level of detail for different EDA stages;
- Effective means for compilation of assumptions and constraints for security schemes, all the way from the system level down to the "bare metal";
- Automated and holistic synthesis of various countermeasures, without inducing negative cross-effects.

We believe that the community is actually well-positioned to address these challenges. EDA tools are driven by metrics and heuristics and also have to tackle trade-offs continuously while step-wise refining the design quality—these principles can certainly be extended towards secure composition of ICs [17].

For any security scheme, it is essential to first define the *threat model*, which describes the adversary's assets, capabilities, constraints, and goals, along with the proposed countermeasures. Depending on the type and time of attack, doing so can become quite complex; e.g., for SCAs and FIAs at runtime, many physical aspects come into play, like means of fault injection, temperature, voltage, power-supply impedance, glitches, etc. Even once threat models are defined properly—and translated into specific metrics and countermeasures which can be handled by EDA tools—they can still have significant weak spots. For one, it is impossible to hinder an adversary from taking further efforts going beyond the modeled means of attack. For another, incorporating the threat model into the EDA tools will be subject to inaccuracies, not only due to limitations on computational cost when exploring the design space of complex ICs with all regular components and various security features, but also due to limitations of metrics and evaluation techniques themselves.

It is understood that EDA tools cannot provide perfect security but it is an essential task to formulate and explore the practical bounds for security schemes when embedded in hardware. EDA tools should assist the designer with automated integration of security features and countermeasures but also needs to formulate the related limitations and remaining risks clearly, to enable effective risk management.

III. Discussion of Prior Art

Table II summarizes security schemes that could be (and partially are already) supported by EDA tools, categorized into design stages versus threat vectors.[2] Some schemes are based on a "red team versus blue team" approach, i.e., they leverage the relevant attack(s) internally, with the objective to inform the designer how to address remaining vulnerabilities or, more challenging, to demonstrate the absence of vulnerabilities. For example, to demonstrate whether an error-detecting scheme can detect all faults means to search for other (types of) faults that are possibly missed. A different approach is to quantify threats through evaluation of metrics, but without considering an explicit attack scenario. For example, countermeasures against SCAs are often assessed by information leakage via statistical or information-theoretical procedures.

In the following, we provide a brief overview and discussion of prior art for each row of Table II.

A. Security-Driven High-Level Synthesis

High-level synthesis (HLS) allocates IP blocks and functional units, binds tasks to these components, and schedules the task execution. An HLS tool would ideally allocate IP blocks automatically as needed for various security-related tasks: (i) secure random number generators (RNGs) [41] for key generation or masking, (ii) PUFs for circuit identification, authentication, and metering [19], [42], (iii) self-authentication logic [20] or wrapper architectures [43] to complicate insertion of Trojans, (iv) error-detecting or shielding architectures against FIAs [10], [18], etc. Another simple countermeasure against SCAs could be to instruct HLS to randomly flush/overwrite registers holding critical data after their use.

While there are works on automated synthesis of masking for software [44], EDA-centric approaches for hardware are still in development. Towards this end, formalized security requirements are an important input for security-centric HLS tools. These can be specified in secure hardware languages like Caisson [45] or SecVerilog [46]. Another language called QIF-Verilog [47] provides the techniques of quantitative information flow (QIF) tracking [48] in the hardware domain. In general, techniques for information-flow tracking developed in the context of software engineering [49] can also be used to validate the resilience of RTL code resulting from HLS tools [14]. The method reported in [49] leverages approximate model counting in order to handle large program state spaces, a concept which is also useful in the context of information-flow tracking for practical EDA use cases.

[2]There is a significant number of studies for most of the table's entries, but we can focus only on selected works within the page limits.

TABLE II
SECURITY SCHEMES SUITABLE FOR INCORPORATION INTO EDA TOOLS

Design Stage	Threat Vectors			
	Side-Channel Attacks	**Fault-Injection Attacks**	**IP Piracy and Counterfeiting**	**Trojans**
High-level synthesis	Information-flow tracking [14]; Integration of masking [5]; Register flushing	Error-detecting architectures [10]; Infective countermeasures [18]	Metering IP (including PUFs) [19]	Self-authentication [20]
Logic synthesis	Gate-level protections [21]; Identification of leaking gates	Automatic fault analysis [22]	Camouflaging [23]; Logic locking [24]	Automatic insertion of security monitors [25]
Physical synthesis (place and route)	Low-level information leakage analysis (TVLA [16], etc.)	Embedding sensors [9], [26]; Shielding [29]	Split manufacturing [27]; Entropy primitives [30]	Embedding sensors [28]
Functional validation	Identification of architectural covert channels [31]	Validation of error-detection properties [32]	Correctness of locked logic; De-obfuscation attacks [33]	Proof-carrying hardware [34]
Timing and power verification	Pre-silicon power/timing simulation [36], [37]	Detailed modeling of fault injections [38]	Validation of low-level properties of PUFs	Fingerprinting [35]
Testing (ATPG, DFT, BIST)	Securing DFT against read-out (scan-chain attacks [39], etc.)	DFX architecture to handle malicious/natural failures	IP protection integrated into DFX infrastructure	Pattern generation for Trojan detection [40]

B. Security-Driven Logic Synthesis

Logic synthesis is the step of compiling the high-level RTL into an actual netlist, mapping it to the technology of choice. This can be combined with gate-level security schemes, e.g., to reduce information leakage exploited by SCAs following the wave dynamic differential logic (WDDL) paradigm [21]. Such "hiding" schemes represent alternatives or complements to formal masking approaches. Methods for automatic fault analysis, some also suitable for logic synthesis, are reviewed in [22]. Moreover, logic synthesis can be tasked to instantiate security monitors to help detecting Trojans at runtime [25].

Concerning IP protection, two approaches are applicable here, camouflaging and logic locking. Logic synthesis has to employ camouflaging according to the scale desired by the designer, where synthesis is constrained to the Boolean functionalities covered by the multi-functional but obfuscated primitives—this is similar to regular but constrained synthesis and is well supported. For locking, however, there is a need to support security requirements formulated at the behavioral level. Currently, locking is implemented directly at the gate-level netlist. Similar to the example for private circuits in Sec. II-B, synthesis is unaware of the security notion for locking. Thus, among others, locking is prone to structural attacks targeting at the synthesized (or layout-level) netlist [50], [51].

C. Security-Driven Physical Synthesis

Physical synthesis is the step of generating an optimized design from the gate-level netlist, through means of place and route (PnR), clock-tree design, timing closure, etc.

Forming the key step towards the "bare metal," it is crucial that physical synthesis considers security notions that are primarily subject to physical phenomena. For example, an important task here is to quantify the information that is leaked through the various side-channels of an IC. The most relevant approach for such evaluation is TVLA [16]. In general, TVLA uses *Welch's t-test statistics* to quantify the differences between the means of two data sets describing some physical phenomena. The validity of TVLA for evaluating SCA resilience is subject to the assumptions made in the threat model, like the noise distribution assumed for the measurements taken by the attacker. Information-theoretic procedures can bound that error using fewer statistical assumptions, but they require careful characterization of the side-channel probability distribution, which is computationally costly (since Maxwell's equations are to be tackled). In any case, most of these metrics and procedures are challenged by the fact that information leakage is multi-dimensional and multi-variate.

Physical design could also be tailored to employ security primitives like RNGs [41] or PUFs [19], [42], shields to protect against FIAs [29], sensors to detect FIAs [9], [26], or Trojan detection circuitry [28]. Since the entropy harnessed by on-chip RNGs and PUFs comes from physical circuit structures, layout-level optimization of their properties is required [30].

As for IP protection, split manufacturing is to be supported at this stage. The security promise of split manufacturing—foremost to hinder IP piracy, but also Trojan insertion, both conducted by foundry adversaries—relies on providing a "meaningless sea of gates with dangling wires" to the untrusted foundry (the higher metal layers are manufactured subsequently by another, trusted facility). Classical EDA flows work holistically on the IC stack, leaving layout-level hints for adversaries, e.g., equipped with machine learning [52]. Thus, it is essential for split manufacturing that physical synthesis is tailored to dissolve such hints (yet optimize for PPA). This can be achieved, e.g., by selective "pushing" of wires to the higher metal layers [53] or by placement perturbation [54].

D. Security-Driven Functional Validation

Validation covers simulation and formal techniques, including equivalence and property checking. Especially the latter is helpful for analysis of security properties and proving their effectiveness. For one, a recent study uses formal methods to identify architectural vulnerabilities in advanced microprocessors [31]. For another, when verifying an error-detection architecture, i.e., when checking for fault coverage, formal analysis developed for transient faults can play a role [32].

For IP protection, verification serves to check the correctness of logic modifications introduced by locking or camouflaging. More importantly even, verification can be used to mimic attackers leveraging satisfiability-based tools (i.e.,

SAT and SMT solvers), and to demonstrate the resilience of protected ICs against such powerful attacks [33].

Finally, concerning Trojans, security properties can be embedded directly in the HDL/RTL to obtain "proof-carrying hardware" [34]. Such schemes should be integrated into property-checker flows and tools.

E. Security-Driven Timing and Power Verification

Timing and power verification serves to achieve design closure. One distinguishes between simulation of timing/power artifacts and "vectorless" analytical approaches, e.g., proving that IR-drop will not exceed a given limit. Simulation approaches are particularly useful for analysis of information leakage through side channels; it is desirable to identify such leakage (or demonstrate its absence) through pre-silicon simulations rather than belatedly measure the final, manufactured ICs. Pre-silicon simulations may also point to the origin of information leakage in the circuitry, thus enabling the designer to fix the underlying problem.

Existing simulation tools work on different abstraction levels and support different degrees of accuracy, from detailed SPICE analysis to fast gate-level approaches. A critical detail for simulation is timing and gate delays; it has been reported that glitches (i.e., transient signals within a clock cycle) influence information leakage [55]. Whether glitches remain present in the actual IC, however, depends on physical-synthesis results, manufacturing variability, and also ambient conditions. It seems an open question how accurate the timing/power models used for simulation must be to obtain reliable prediction about the expected information leakage.

Timing/power verification is also the stage to run detailed analysis of fault injections using accurate electrical models [38], or to verify the behavior of PUFs in terms of entropy, reliability, and uniqueness. Simulation also serves well for *fingerprinting* [35], a countermeasure against Trojans, which is based on checking consistency of path delays.

F. Security-Driven Testing

Testability is crucial, yet contradictory to security to some degree [56]. That is because test, diagnosis, and debug features enable comprehensive access to IC internals, providing an attacker the opportunity to read out sensitive information (e.g., via scan-based attacks [39]). As a consequence, emerging test standards will also incorporate security measures [57].

More complex ICs incorporate a "design for X" (DFX) infrastructure, which combines classical scan-based testing with build-in self test (BIST) features for logic and memory, transient-fault detection and re-configuration logic, circuitry for yield management, and debug and diagnostic features [58]. To integrate FIA detection into the same DFX infrastructure seems only logical. However, distinguishing between natural and malicious faults is non-trivial [59], and the responses should be different: fastest possible recovery and resumption of regular operation upon a natural fault, but re-keying or even discontinuation of service upon a tampering attempt. Therefore, future security-aware DFX infrastructures should enable such distinction. Besides, they may also manage IP protection techniques, e.g., for key management for locking.

There is extensive prior art for detecting Trojans through means of testing. This covers (i) functional tests that aim at triggering Trojans [40] and (ii) parametric tests that aim at detecting Trojans' fingerprints through side-channels [60]. While such tests can be included into automatic test pattern generation (ATPG) tools, their effectiveness in reliably identifying strategically hidden Trojans in large and complex ICs remains to be proven.

IV. Strategies and Challenges Towards Secure Composition Using EDA Tools

Security of any system is subject to its weakest link, and ICs form no exception here. We have covered, in overview, the large variety of hardware-related threats and countermeasures, along with some discussion of current limitations.

It is known that not all types or implementations of countermeasures are *composable*, e.g., adding error-detecting logic can deteriorate resilience against SCAs [61]. Thus, tools for joint compilation of countermeasures and, even more importantly, for verifying their effectiveness are required. Ideally, once the security-enforcing designers have implemented yet another countermeasure, they can re-run the envisioned security-centric EDA flow which then covers all threats, also seemingly unrelated ones, to hinder that any countermeasure has become inadvertently compromised.

To become a reality, such security-centric EDA tools require effective and efficient security metrics and evaluation techniques. The whole EDA domain is metrics-driven, and EDA tools are well positioned to balance between, e.g., a circuit's area and testability, all quantified by meaningful metrics.

While several security metrics and evaluation techniques are known [12], as also outlined in this paper, the necessary assumption of an intelligent and strategic attacker complicates their definition and usage. For example, a transient fault that leads to a critical system failure can be ignored during reliability analysis in case it is extremely unlikely to occur. When it comes to resistance against FIAs, however, the attacker may put extra effort into injecting precisely this fault; it cannot be ignored anymore. Having to account for such "unlikely but possible" events poses a significant burden for security analysis and on appropriate strategies to incorporate such analysis into EDA tools. This also implies that one can expect some security metrics to act more like step functions, where certain efforts must be spent to reach a security level, but spending more will not provide additional benefits. This is fundamentally different from classical metrics like area or power consumption and should be considered accordingly for security-aware design space exploration.

V. Conclusion

EDA tools are traditionally a key enabler for complex ICs and electronic systems. Nowadays, an increasing number of applications becomes security-critical and ICs must offer protection against hardware-oriented attacks, yet the support

by EDA tools is lacking for this matter. We outlined short- to medium-term potentials for security-driven design methods to be integrated into EDA tools. We also identified conceptual challenges for secure composition of countermeasures against various threat vectors and for security metrics.

Acknowledgements

This work originates from Dagstuhl Seminar 19301, *Secure Composition for Hardware Systems*, July 21–26, 2019.

References

[1] E. Brier *et al.*, "Correlation Power Analysis with a Leakage Model," *CHES*, vol. 3156, pp. 16–29, 2004.

[2] A. Barenghi *et al.*, "Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.

[3] P. Kocher *et al.*, "Spectre Attacks: Exploiting Speculative Execution," *SP*, vol. 1, pp. 19–37, 2019.

[4] D. Bellizia *et al.*, "Secure Double Rate Registers as an RTL Countermeasure Against Power Analysis Attacks," *TVLSI*, vol. 26-7, 2018.

[5] H. Groß *et al.*, "Domain-Oriented Masking: Compact Masked Hardware Impl. with Arbitrary Protection Order," in *TIS@CCS*. ACM, 2016.

[6] B. Selmke *et al.*, "Attack on a DFA Protected AES by Simultaneous Laser Fault Injections," in *FDTC*, 2016, pp. 36–46.

[7] A. Dehbaoui *et al.*, "Injection of transient faults using electromagnetic pulses Practical results on a cryptographic system," in *ePrint-123*, 2012.

[8] V. van der Veen *et al.*, "Drammer: Deterministic Rowhammer Attacks on Mobile Platforms," in *CCS*, 2016, pp. 1675–1689.

[9] G. D. Natale *et al.*, "Hidden-Delay-Fault Sensor for Test, Reliability and Security," in *DATE*. IEEE, 2019, pp. 316–319.

[10] B. Karp *et al.*, "Security-oriented Code-based Architectures for Mitigating Fault Attacks," in *DCIS*. IEEE, 2018, pp. 1–6.

[11] J. Knechtel *et al.*, "Protect your chip design intellectual property: An overview," in *COINS*, 2019, pp. 211–216.

[12] M. Rostami *et al.*, "A Primer on Hardware Security: Models, Methods, and Metrics," *JProc*, vol. 102, no. 8, pp. 1283–1295, 2014.

[13] S. Bhunia *et al.*, Eds., *The Hardware Trojan War: Attacks, Myths, and Defenses*. Springer, 2018.

[14] C. Pilato *et al.*, "TaintHLS: High-Level Synthesis for Dynamic Information Flow Tracking," *IEEE TCAD*, vol. 38-5, p. 798, 2019.

[15] D. B. Roy *et al.*, "From Theory to Practice of Private Circuit: A Cautionary Note," in *ICCD*, 2015, pp. 296–303.

[16] J. Cooper *et al.*, "Test Vector Leakage Assessment (TVLA) Methodology in Practice," in *International Cryptographic Module Conference*, 2013.

[17] P. Ravi *et al.*, "Security is an architectural design constraint," *MICPRO*, vol. 68, pp. 17 – 27, 2019.

[18] S. Patranabis *et al.*, "Fault Tolerant Infective Countermeasure for AES," *J. Hardware and Systems Security*, vol. 1, no. 1, pp. 3–17, 2017.

[19] Y. Alkabani *et al.*, "Active Hardware Metering for Intellectual Property Protection and Security," in *USENIX Security Symposium*, 2007.

[20] K. Xiao *et al.*, "BISA: Built-in Self-authentication for Preventing Hardware Trojan Insertion," in *HOST*. IEEE, 2013, pp. 45–50.

[21] K. Tiri *et al.*, "A Digital Design Flow for Secure Integrated Circuits," *IEEE T.CAD of ICs and Systems*, vol. 25, no. 7, pp. 1197–1208, 2006.

[22] J. Breier *et al.*, Eds., *Automated Methods in Cryptographic Fault Analysis*. Springer, 2019.

[23] J. Rajendran *et al.*, "Security Analysis of Logic Obfuscation," in *DAC*. ACM, 2012, pp. 83–89.

[24] J. A. Roy *et al.*, "EPIC: Ending Piracy of Integrated Circuits," in *DATE*. ACM, 2008, pp. 1069–1074.

[25] T. F. Wu *et al.*, "TPAD: Hardware Trojan Prevention and Detection for Trusted Integrated Circuits," *TCAD*, vol. 35, no. 4, pp. 521–534, 2016.

[26] M. Khairallah *et al.*, "DFARPA: Differential fault attack resistant physical design automation," in *DATE*, 2018, pp. 1171–1174.

[27] C. McCants, "Trusted integrated chips (TIC) program," IARPA, Tech. Rep., 2016. [Online]. Available: https://www.ndia.org/-/media/sites/ndia/meetings-and-events/divisions/systems-engineering/past-events/trusted-micro/2016-august/mccants-carl.ashx

[28] X. Zhang *et al.*, "Detection of Trojans Using A Combined Ring Oscillator Network and Off-chip Transient Power Analysis," *JETC*, vol. 9, no. 3, pp. 25:1–25:20, 2013.

[29] H. Li *et al.*, "Security Evaluation At Design Time Against Optical Fault Injection Attacks," *IEE Proc.-Inf. Security*, vol. 153-1, pp. 3–11, 2006.

[30] Y. Guo *et al.*, "Variation Enhancement of Arbiter PUFs with Asymmetric Layout," in *MWSCAS*. IEEE, 2018, pp. 841–844.

[31] M. R. Fadiheh *et al.*, "Processor Hardware Security Vulnerabilities and their Detection by Unique Program Execution Checking," in *DATE*. IEEE, 2019, pp. 994–999.

[32] G. Fey *et al.*, "Effective Robustness Analysis Using Bounded Model Checking Techniques," *TCAD*, vol. 30, no. 8, pp. 1239–1252, 2011.

[33] K. Z. Azar *et al.*, "SMT Attack: Next Generation Attack on Obfuscated Circuits with Capabilities and Performance Beyond the SAT Attacks," *IACR TCHES*, vol. 2019, no. 1, pp. 97–122, 2019.

[34] E. Love *et al.*, "Proof-Carrying Hardware Intellectual Property: A Pathway to Trusted Module Acquisition," *IEEE Trans. Information Forensics and Security*, vol. 7, no. 1, pp. 25–40, 2012.

[35] Y. Jin *et al.*, "Hardware Trojan Detection Using Path Delay Fingerprint," in *HOST*. IEEE Computer Society, 2008, pp. 51–57.

[36] J. Jiang *et al.*, "MIRID: Mixed-Mode IR-Drop Induced Delay Simulator," in *Asian Test Symposium*. IEEE ComSoc, 2013, pp. 177–182.

[37] M. T. He *et al.*, "RTL-PSC: Automated power side-channel leakage assessment at register-transfer level," *arXiv*, 2019.

[38] J. Dutertre *et al.*, "Laser Fault Injection at the CMOS 28 nm Technology Node: an Analysis of the Fault Model," in *FDTC*. IEEE ComSoc, 2018.

[39] B. Yang *et al.*, "Secure Scan: A Design-for-test Architecture for Crypto Chips," in *DAC*. ACM, 2005, pp. 135–140.

[40] R. S. Chakraborty *et al.*, "MERO: A Statistical Approach for Hardware Trojan Detection," in *CHES*, vol. 5747. Springer, 2009, pp. 396–410.

[41] W. Schindler, "Random Number Generators for Cryptographic Applications," in *Cryptographic Engineering*. Springer, 2009, pp. 5–23.

[42] U. Rührmair *et al.*, "PUFs At A Glance," in *DATE*, 2014, pp. 1–6.

[43] A. Basak *et al.*, "Security Assurance for System-on-Chip Designs With Untrusted IPs," *IEEE TIFS*, vol. 12, no. 7, pp. 1515–1528, 2017.

[44] H. Eldib *et al.*, "Synthesis of Masking Countermeasures Against Side Channel Attacks," in *CAV*, vol. 8559. Springer, 2014, pp. 114–130.

[45] X. Li *et al.*, "Caisson: A Hardware Description Language for Secure Information Flow," in *ACM SIGPLAN PLDI*, 2011, pp. 109–120.

[46] D. Zhang *et al.*, "A Hardware Design Language for Timing-Sensitive Information-Flow Security," *SIGPLAN*, vol. 50-4, pp. 503–516, 2015.

[47] X. Guo *et al.*, "QIF-Verilog: Quantitative Information-Flow based Hardware Description Languages for Pre-Silicon Security Assessment," in *HOST*. IEEE, 2019, pp. 91–100.

[48] P. Mardziel *et al.*, "Quantifying Information Flow for Dynamic Secrets," in *IEEE SP*, 2014, pp. 540–555.

[49] F. Biondi *et al.*, "Scalable Approximation of Quantitative Information Flow in Programs," in *Proc. of VMCAI Conference*, 2018, pp. 71–93.

[50] P. Chakraborty *et al.*, "SAIL: Machine Learning Guided Structural Analysis Attack on Hardware Obfuscation," in *AHOST*, 2018, pp. 56–61.

[51] F. Yang *et al.*, "Stripped Functionality Logic Locking with Hamming Distance Based Restore Unit (SFLL-hd) – Unlocked," *TIFS*, 2019.

[52] H. Li *et al.*, "Attacking Split Manufacturing from a Deep Learning Perspective," in *DAC*, 2019, pp. 135:1–135:6.

[53] S. Patnaik *et al.*, "Concerted wire lifting: Enabling secure and cost-effective split manufacturing," in *ASPDAC*, 2018, pp. 251–258.

[54] A. Sengupta *et al.*, "Rethinking Split Manufacturing: An Information-theoretic Approach with Secure Layout Techniques," in *ICCAD*. IEEE, 2017, pp. 329–326.

[55] T. Moos *et al.*, "Glitch-Resistant Masking Revisited or Why Proofs in the Robust Probing Model are Needed," *IACR TCHES*, vol. 2019, no. 2, pp. 256–292, 2019.

[56] I. Polian, "Hardware Security and Test: Friends or Enemies?" *it - Information Technology*, vol. 56, no. 4, pp. 192–202, 2014.

[57] E. Valea *et al.*, "A Survey on Security Threats and Countermeasures in IEEE Test Standards," *IEEE Des. & Test*, vol. 36-3, pp. 95–116, 2019.

[58] I. Parulkar *et al.*, "DFX of a 3rd generation, 16-core/32-thread ultrasparc- CMT microprocessor," in *ITC*. IEEE, 2008, pp. 1–10.

[59] B. Karp *et al.*, "Detection and Correction of Malicious and Natural Faults in Cryptographic Modules," in *PROOFS*, vol. 7, 2018, pp. 68–82.

[60] J. Aarestad *et al.*, "Detecting Trojans Through Leakage Current Analysis Using Multiple Supply Pad IDDQS," *IEEE Trans. Information Forensics and Security*, vol. 5, no. 4, pp. 893–904, 2010.

[61] F. Regazzoni *et al.*, "Interaction Between Fault Attack Countermeasures and the Resistance Against Power Analysis Attacks," in *Fault Analysis in Cryptography*, ser. Inf. Sec. and Crypt. Springer, 2012, pp. 257–272.