Dominik Sisejkovic
Rainer Leupers

# Logic Locking

## A Practical Approach to Secure Hardware

Springer

# Logic Locking

Dominik Sisejkovic • Rainer Leupers

# Logic Locking

## A Practical Approach to Secure Hardware

Springer

Dominik Sisejkovic
RWTH Aachen University
Aachen, Germany

Rainer Leupers
RWTH Aachen University
Aachen, Germany

ISBN 978-3-031-19122-0 ISBN 978-3-031-19123-7 (eBook)
https://doi.org/10.1007/978-3-031-19123-7

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

A subtle change that leads to disastrous consequences—hardware Trojans undoubtedly pose one of the greatest security threats to the modern age. How to protect hardware against these malicious modifications? One potential solution hides within logic locking, a prominent hardware obfuscation technique. In this book, we take a step-by-step approach to understanding logic locking, from its fundamental mechanics, over the implementation in software, down to an in-depth analysis of security properties in the age of machine learning. This book can be used as a reference for beginners and experts alike who wish to dive into the world of logic locking, thereby having a holistic view of the entire infrastructure required to design, evaluate, and deploy modern locking policies.

Aachen, Germany — Dominik Sisejkovic
Aachen, Germany — Rainer Leupers
August 2022

# Contents

# Acronyms

| | |
|---|---|
| 3PIP | Third-Party Intellectual Property |
| AES | Advanced Encryption Standard |
| AGR | AppSAT-Guided Removal |
| AIG | And-Inverter Graph |
| ALU | Arithmetic Logic Unit |
| ANN | Artificial Neural Network |
| ANT | AND Netlist Test |
| APD | Area-Power-Delay |
| ASIC | Application-Specific Integrated Circuit |
| AST | Abstract Syntax Tree |
| AT | Area-Timing |
| ATI | AND-Tree Insertion |
| ATPG | Automatic Test Pattern Generation |
| BEOL | Back End of Line |
| BeSAT | Behavioral SAT |
| BFS | Breadth-First Search |
| BISA | Built-In Self-Authentication |
| C1HT | Class-1 Hardware Trojan |
| C2HT | Class-2 Hardware Trojan |
| CAA | Cycle Analysis Attack |
| CLC | Control-Lock Circuit |
| CLIC-A | Characterization of Locked Integrated Circuits via ATPG |
| CMOS | Complementary Metal-Oxide Semiconductor |
| CNN | Convolutional Neural Network |
| CPU | Central Processing Unit |
| D-MUX | Deceptive Multiplexer Logic Locking |
| DAG | Direct Acyclic Graph |
| DC | Design Compiler |
| DDIP | Double DIP |
| DFA | Differential Fault Analysis |
| DfTr | Design for Trust |

| | |
|---|---|
| DIP | Distinguishing Input Pattern |
| DLCL | DIP Learning on CAS-Lock Attack |
| DoS | Denial of Service |
| DPA | Differential Power Analysis |
| eD-MUX | Enhanced D-MUX |
| EDA | Electronic Design Automation |
| FEOL | Front End of Line |
| FLL | Fault Analysis-Based Logic Locking |
| FPGA | Field-Programmable Gate Array |
| GA | Genetic Algorithm |
| gD-MUX | Generalized D-MUX |
| GE | Gate Equivalent |
| GNN | Graph Neural Network |
| GSS | Generalized Set Scenario |
| HCA | Hill-Climbing Attack |
| HD | Hamming Distance |
| HDL | Hardware Description Language |
| HLS | High-Level Synthesis |
| HT | Hardware Trojan |
| HW | Hardware |
| IC | Integrated Circuit |
| IFS | Identify Flip Signal |
| ILC | Inter-Locking Circuit |
| IO | Input and Output |
| IP | Intellectual Property |
| ISA | Instruction Set Architecture |
| KBM | Key-Bit Mapping |
| KPA | Key Prediction Accuracy |
| LC | Layout Camouflaging |
| LCALL | Logic Cone Analysis Logic Locking |
| LCBFA | Logic Cone-Based Brute-Force Attack |
| LCSBLL | Logic Cone Size-Based Logic Locking |
| LUT | Lookup Table |
| LVE | Locality Vector Extraction |
| MiG-V | Made in Germany RISC-V |
| ML | Machine Learning |
| MLP | Multi-Layer Perceptron |
| MUX | Multiplexer |
| NAS | Neural Architectural Search |
| OCP | Optical Contactless Probing |
| OG | Oracle-Guided |
| OL | Oracle-Less |
| OOP | Object-Oriented Programming |
| PF | Point Function |
| PSA | Path-Sensitization Attack |

| | |
|---|---|
| PSO | Particle Swarm Optimization |
| RAA | Rationality Analysis Attack |
| RARLL | Redundancy Attack Resistant Logic Locking |
| RE | Reverse Engineering |
| RLL | Random Logic Locking |
| RNT | Random Netlist Test |
| RSAS | Robust SAS |
| RTL | Register-Transfer Level |
| SAAM | Structural Analysis Attack on MUX-Based Locking |
| SARO | Scalable Attack-Resistant Obfuscation |
| SAS | Strong Anti-SAT |
| SAT | Boolean Satisfiability Problem |
| SFLL | Stripped-Functionality Logic Locking |
| SGS | Sensitization-Guided SAT |
| SKRA | Statistical Key Recovery Attack |
| SLL | Strong (Secure) Logic Locking |
| SMT | Satisfiability Modulo Theory |
| SoC | System-on-Chip |
| SPS | Signal Probability Skew |
| SRS | Self-Referencing Scenario |
| TAAL | Tampering Attack on Any Key-Based Logic Locked Circuit |
| TAL | Test-Aware Locking |
| TDM | Test-Data Mining |
| TGA | Topology-Guided Attack |
| TGARLL | TGA-Resistant Logic Locking |
| TRENTOS | Trusted Entity Operating System |
| TRLL | Truly Random Logic Locking |
| TTLock | Tenacious and Traceless Logic Locking |

# Notation

| | |
|---|---|
| $C^f$ | ($C^f$) Binary Galois field 2 |
| $D^f$ | Functional deceptiveness factor |
| $FHS$ | Functional hardware security |
| $G^f$ | Golden factor |
| $H$ | Frequency histogram |
| $IC_{ll}$ | Logic-locked design |
| $IC$ | Original (unlocked) design |
| $I$ | A single input pattern |
| $KSS$ | Key-space size |
| $K$ | Activation key |
| $O$ | A single output pattern |
| $SHS_{cc}$ | Structural complexity change |
| $SHS_{en}$ | Structural key-gate entropy |
| $SHS$ | Structural hardware security |
| $S^f$ | Secrecy factor |
| $TT$ | Truth table |
| $\mathcal{I}$ | Set of all input patterns |
| $\mathcal{O}$ | Set of all output patterns |
| $d_e$ | Euclidian distance |
| $f^{dc}$ | Key-space decrease function |
| $f^{kd}$ | Key-gate distribution measure function |
| $s_d$ | Structural distribution vector |
| $s_{max}$ | Maximal structural distance vector |
| $s_{opt}$ | Optimal structural distribution vector |
| $v_{ext}$ | Extracted feature vector |
| $v_{max}$ | Maximal distance feature vector |
| $v_{opt}$ | Optimal feature vector |

# Part I
# Hardware Security and Trust: Threats and Solutions

# Chapter 1
# Introduction


Computer security has become a driving force in the design of modern electronics systems. Over many years, security primitives, specifically in software, have been extensively researched. Hardware (HW) security, in comparison, is a relatively young field, since HW has been traditionally considered immune to attacks, representing a root of trust for any electronic system. However, over the last three decades, an increasing number of vulnerabilities have been identified with the root cause in the hardware itself [29]. Attacks that exploit these vulnerabilities can be broadly separated into two categories. The first category encompasses all attack vectors that are enabled due to an overlooked construction fault in the HW implementation, opening the door for a range of attack vectors, predominantly in the form of side-channel attacks and exploits of other unintended HW side effects. Notable examples in the last years include transient execution attacks, such as Meltdown [106] and Spectre [93], as well as security exploits in dynamic random-access memories, such as RowHammer [92, 118]. The second category includes more recent attack types that are enabled by intentional, malicious changes in the HW, commonly known as Hardware Trojans (HTs) [30]. The challenges introduced by these modifications have a deep impact on the research and development landscape of hardware security, particularly as they can serve as key enablers of a theoretically unlimited attack surface; including information leakage, reliability degradation, and denial of service, among others.

The introduction of HTs yields an interesting question: what is their root cause? Nowadays, a highly competitive environment, short time-to-market, and the ever-increasing need for reduced design and production costs have transformed the Integrated Circuit (IC) supply chain into a global effort, driven by third-party Intellectual Property (IP), subcontracting external design houses, and outsourcing the fabrication to off-site foundries. This deep reliance on external parties has led to a far-reaching consequence—the loss of trust and assurance. Hence, legitimate IP owners are faced with the possibility of injected HTs, leading to untrustworthy HW components. And this challenge is, by all means, a serious one. A wide range

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

D. Sisejkovic, R. Leupers, *Logic Locking*,
https://doi.org/10.1007/978-3-031-19123-7_1

of engineers are involved in the design and production of HW, thereby having full access to a design and often operating across multiple organizations, countries, and even continents. It only takes a single rogue entity, implanting a tiny, stealthy, and carefully placed modification, to lay the foundation for a catastrophic attack. This covert nature of HTs makes it difficult to catch them in the wild, in particular, due to the inherent complexity of modern circuits and shrinking feature sizes. Thus, only a handful of alleged HTs have been reported. For example, more than a decade ago, a Syrian radar system failed to warn of an incoming airstrike, reportedly because of HTs embedded in the defense systems [2, 114]. Even though it is difficult to verify the inclusion of HTs in such incidences, the very potential of this tiny, malicious design modification has become a focal point within research and industry. The US military and intelligence executives have placed hardware Trojans among the most severe threats the nation might face in the event of war [111]. Moreover, the US Defense Advanced Research Project Agency (DARPA) has issued multiple funding programs to address the issue of trustworthy electronics, including the TRUST [48], IRIS [46], and SHIELD [47] program, among others. The seriousness of this issue has also been recognized within Germany. The German Federal Ministry of Education and Research (BMBF) has issued a framework program for 2021-2024 to tackle the challenges of trustworthy and sustainable microelectronics for Germany and Europe [32] with a range of projects already underway [33].

The efforts in mitigating HTs evolve around two focal points: detection and prevention. Trojan detection aims at detecting and removing potential HTs, possibly before these are placed in silicon. However, detection approaches are still far from a complete solution due to multiple reasons. First, HTs can be injected on many different levels of the HW design abstraction and in various stages of the IC supply chain. This makes it challenging to derive an effective detection mechanism. Second, even if comprehensive (and often destructive) reverse engineering procedures are deployed to verify the absence of Trojans in chips after production, this does not guarantee that all produced ICs are HT-free. Therefore, more focus has been given to preventing HT insertion by design. In particular, logic locking has evolved as a premier technique to protect against HT insertion by means of key-controlled functional and structural design changes that aim at protecting the asset—the HW design—throughout the IC supply chain [218]. Hereby, the defensive mechanism is built on the assumption that an attacker is required to perform extensive reverse engineering to insert and construct an intelligible, design-specific hardware Trojan. Hence, the locking-induced changes increase the complexity of the attack by binding the correct behavioral and structural HW characteristics to a secret key. Nevertheless, the evolutionary timeline of logic locking has been riddled with a wide range of attack vectors and unclear security objectives. This has led to logic locking largely remaining a theoretical concept without any tangible outcome.

To address this issue, in this book, we aim at closing the practicality gap in logic locking by devising a set of models, software tools, attacks, and schemes that enable the evaluation and application of logic locking to complex, silicon-proven HW designs within a concise and realistic attack scenario [164].

## 1.1 Outline

This book is organized into four parts covering eleven chapters. The structure is meant to guide the reader from basic concepts on hardware security to software implementation details for silicon-ready logic locking. By the end of the book, readers should be able to understand how logic locking operates and how it can be challenged, how to implement the right tools to deploy locking schemes, and finally, how to evaluate the security of logic locking with emerging machine learning-based approaches. The book is structured as follows.

**Background** First, preliminaries on electronics supply chain threats and solutions are presented in Chap. 2.

**Hardware Trojans** Chapter 3 introduces the anatomy of Hardware Trojans alongside existing classification systems. Moreover, a consolidated classification is introduced that considers the impact of defensive approaches. Finally, the chapter compares the effectiveness of existing Trojan-insertion countermeasures w.r.t the constructed classification.

**Working Principles and Attack Scenarios** The mechanics of logic locking, its impact on reverse engineering as well as common attack scenarios are discussed in Chap. 4.

**Attacks and Schemes** An overview and classification of deobfuscation attacks and logic locking schemes is presented in Chap. 5.

**Security Metrics** Chapter 6 introduces the design of one of the first generalized hardware security metrics with respect to logic locking. Furthermore, based on the introduced concepts, the security–cost trade-off problem is analyzed through a case study that evaluates the impact of a higher cost budget on the security properties of logic locking.

**Software Framework** The design and implementation of a software-based logic locking framework for the protection of complex multi-module HW designs is discussed in Chap. 7. The framework is designed in the form of an end-to-end locking procedure, featuring a technology-independent design representation and an extensible code base for rapid scheme prototyping. Furthermore, the constructed framework ensures the deployment of logic locking within an industry-ready setting without impacting the traditional design, verification, and fabrication steps.

**Processor Integrity Protection** The implementation of framework extensions in the form of two protection schemes, Inter-Lock and Control-Lock, is presented in Chap. 8. Inter-Lock embodies a cross-module, logic locking meta-scheme that scales any locking policy across multiple HW modules, thereby creating additional functional and structural dependencies between the selected components.

This cross-module policy is the first to widen the security implications of logic locking to complex hardware designs. Control-Lock implements an inter-module encryption mechanism that aims at protecting critical HW control signals against the exploitation by software-controlled hardware Trojans. The impact of both procedures is evaluated on silicon-proven RISC-V processor cores. Finally, the presented research developments are successfully transferred to industry, resulting in the first comprehensively logic-locked and commercially available processor—the "Made in Germany RISC-V" (MiG-V) core [195]. Hereby, a major milestone is achieved in the domain of logic locking.

**Security Evaluation with Machine Learning** The introduction of fundamental concepts in attacks and defenses in logic locking with respect to Machine Learning (ML) [170] is presented in Chap. 9. First, SnapShot is presented; an attack that utilizes artificial neural networks to directly predict correct key bits from a locked netlist. Furthermore, a neuroevolutionary procedure is developed to automatically assemble suitable neural architectures for the selected prediction problem. Furthermore, the generalized set and self-referencing attack scenario are discussed as standard attack vectors in a machine learning-based setting.

**Designing Deceptive Logic Locking** Based on the lessons learned from Chap. 9, the first theoretical test for uncovering structural leakage points is introduced in Chap. 10. The test embodies a procedure that can lead to the identification of fundamental security vulnerabilities that are exploitable by ML-driven attacks. The analysis results are used as a basis to construct a multiplexer-based locking policy that targets learning resilience. Through further evaluation steps, an analysis of challenges in ML-resilient locking is performed. Furthermore, a novel attack is presented, uncovering a major fallacy in existing multiplexer-based schemes. The introduced concepts, policies, and attacks offer the potential to establish the cornerstones for the design of next-generation logic locking in the era of machine learning.

**Next Steps** New research directions and open challenges are discussed in Chap. 11. Finally, Chap. 12 concludes the book.

# Chapter 2
# Background


To better understand the contents of this book, this chapter introduces the following preliminaries. An overview of the major security vulnerabilities in the electronic supply chain is presented in Sect. 2.1. Prominent Design for Trust (DfTr) solutions are detailed in Sect. 2.2. Finally, Sect. 2.3 concludes the chapter.

## 2.1 Electronics Supply Chain Threats

The complexity and distributed nature of the modern electronics supply chain have led to a lack of trust and assurance thereof. Consequently, a range of attack vectors has been introduced to steal, illegitimately sell, or compromise the integrity of Integrated circuits (ICs). The following subsections present the background on selected trust issues. More details can be found in [29].

### *2.1.1 Reverse Engineering*

In the context of hardware, Reverse Engineering (RE) is defined as the process of extracting a set of specifications for a hardware design by someone other than the original design owner [136]. Hereby, RE can be deployed at different circuit abstraction levels [125]. The legitimacy of RE depends on what its result is used for. Thus, the product of RE can be utilized for either verification purposes or illegal actions, such as hardware Trojan insertion or Intellectual Property (IP) theft.

The process of reverse engineering hardware designs includes a set of manual and semi-automated steps [20, 63, 176, 187, 198]. Starting from a fabricated IC, the RE flow can be divided into netlist extraction and functionality identification. The former extracts a netlist representation of the physical chip through multiple

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

D. Sisejkovic, R. Leupers, *Logic Locking*,
https://doi.org/10.1007/978-3-031-19123-7_2

successive steps, including the sample preparation (package removal and delayering), image acquisition, layout extraction, and netlist generation. The latter concerns the acquisition of a high-level description of the intended functionally of the design [22].

Due to its complexity, the process of reverse engineering has no clear guidelines. Only recently, first attempts to analyze the required cognitive and technical skills to perform RE have been analyzed [25]. Nevertheless, it still remains a challenge to fully automate the process as well as quantify the complexity of RE for a specific design.

### 2.1.2 Hardware Trojans

Hardware Trojans (HTs) are defined as malicious and intentional circuit modifications that can result in undesired circuit behavior after deployment [30, 201]. The malicious behavior can be manifested in the form of information leakage, performance degradation, increased power dissipation, Denial of Service (DoS) attacks, and others.

The anatomy of hardware Trojans consists of a trigger and a payload [28, 39]. The trigger activates the Trojan based on a specific activation event, such as the occurrence of specific data values or circuit states, external signals, number of cycles, and others. The malicious behavior of the Trojan is manifested in the form of the payload. The malicious circuitry can be inserted into the hardware at different design levels, depending on which entities in the supply chain are considered trustworthy. Thus, in principle, HTs can be introduced into a design during specification, design, fabrication, testing, or assembly and packaging, thereby being initiated by untrusted personnel or Electronic Design Automation (EDA) tools. This broad attack landscape has led to the introduction of many hardware Trojan taxonomies and example implementations [51, 90, 143, 148, 186, 208]. Furthermore, a recent study even demonstrated how hardware Trojans can effortlessly and automatically be implanted in finalized layouts [120].

The diverse design possibilities, insertion locations, and stealthy implementation nature make HT detection a challenging task. Moreover, similar to regular faults, the later Trojans are detected in the design and production flow, the costlier and more difficult it becomes for the IP owner to act. This problem is further exacerbated by the fact that HTs are often assumed to be deployed by untrusted, external foundries—beyond the control of the legitimate IP owner.

Another important aspect lies within the resources required to design, implement, and inject hardware Trojans [72, 154]. In general, *design-independent* HTs can be implemented and inserted with very little knowledge about the design's functionality or structure. These Trojans, however, are likely to be detected, exhibit uncontrollable trigger mechanisms, and result in random payloads, thus mimicking a random fault. In contrast, *design-dependent* Trojans can be constructed to allow for a controllable activation, stealthy implementation, and dedicated payload, thus leading to high-impact attack scenarios. Consequently, design-dependent HTs