Dominik Sisejkovic · Rainer Leupers

Logic Locking

Ein praktischer Ansatz für sichere Hardware

Springer Vieweg

Logic Locking

Dominik Sisejkovic · Rainer Leupers

Logic Locking

Ein praktischer Ansatz für sichere Hardware

Springer Vieweg

Dominik Sisejkovic
RWTH Aachen University
Aachen, Deutschland

Rainer Leupers
RWTH Aachen University
Aachen, Deutschland

ISBN 978-3-031-54399-9 ISBN 978-3-031-54400-2 (eBook)
https://doi.org/10.1007/978-3-031-54400-2

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über https://portal.dnb.de abrufbar.

Planung/Lektorat: Charles Glaser
Springer Vieweg ist ein Imprint der eingetragenen Gesellschaft Springer Nature Switzerland AG und ist ein Teil von Springer Nature.
Die Anschrift der Gesellschaft ist: Gewerbestrasse 11, 6330 Cham, Switzerland

Das Papier dieses Produkts ist recycelbar.

Vorwort

Eine subtile Änderung, die zu katastrophalen Folgen führt – Hardware-Trojaner stellen zweifellos eine der größten Sicherheitsbedrohungen des modernen Zeitalters dar. Wie kann man Hardware gegen diese bösartigen Modifikationen schützen? Eine mögliche Lösung verbirgt sich im Logic Locking, einer prominenten Technik zur Verschleierung von Hardware. In diesem Buch gehen wir schrittweise an das Verständnis von Logic Locking heran, von ihren grundlegenden Mechanismen, über die Implementierung in Software, bis hin zu einer tiefgehenden Analyse der Sicherheitseigenschaften im Zeitalter des maschinellen Lernens. Dieses Buch kann als Referenz für Anfänger und Experten gleichermaßen verwendet werden, die in die Welt des Logic Lockings eintauchen möchten und ermöglicht so einen ganzheitlichen Blick auf die gesamte Infrastruktur, die erforderlich ist, um moderne Verriegelungsmaßnahmen zu entwerfen, zu bewerten und zu implementieren.

Aachen, Deutschland
August 2022

Dominik Sisejkovic
Rainer Leupers

Inhaltsverzeichnis

Abkürzungen

3PIP	Third-Party Intellectual Property; dt. Drittanbieter – Geistiges Eigentum
AES	Advanced Encryption Standard; dt. Erweiterter Verschlüsselungsstandard
AGR	AppSAT-Guided Removal; dt. AppSAT-gesteuerte Entfernung
AIG	AND-Inverter Graph
ALU	Arithmetic Logic Unit; dt. Arithmetisch-logische Einheit
ANN	Artificial Neural Network; dt. Künstliches neuronales Netzwerk
ANT	AND-Netlist-Test
APD	Area-Power-Delay; dt. Flächen-Leistungs-Verzögerung
ASIC	Application-Specific Integrated Circuit; dt. Anwendungsspezifische integrierte Schaltung
AST	Abstract Syntax Tree; dt. Abstrakter Syntaxbaum
AT	Area-Timing; dt. Flächen-Zeitmessung
ATI	AND-Tree Insertion; dt. AND-Baum-Einfügung
ATPG	Automatic Test Pattern Generation; dt. Automatische Testmustererzeugung
BEOL	Back End of Line
BeSAT	Verhaltensbezogener SAT
BFS	Breitensuche
BISA	Integrierte Selbstauthentifizierung
C1HT	Klasse-1 Hardware-Trojaner
C2HT	Klasse-2 Hardware-Trojaner
CAA	Zyklusanalyse-Angriff
CLC	Steuerungs-Sperrschaltung
CLIC-A	Charakterisierung von gesperrten integrierten Schaltkreisen über ATPG
CMOS	Komplementär-Metalloxid-Halbleiter
CNN	Faltendes neuronales Netzwerk
CPU	Zentrale Verarbeitungseinheit
D-MUX	Täuschende Multiplexer-Logikverriegelung

DAG	Direkter azyklischer Graph
DC	Design Compiler
DDIP	Doppel DIP
DFA	Differenzielle Fehleranalyse
DfTr	Design für Vertrauen
DIP	Unterscheidendes Eingabemuster
DLCL	DIP-Lernen bei CAS-Lock-Angriff
DoS	Verweigerung des Dienstes
DPA	Differenzielle Leistungsanalyse
eD-MUX	Verbesserter D-MUX
EDA	Elektronische Designautomatisierung
FEOL	Front End of Line
FLL	Fehleranalyse-basiertes Logikschloss
FPGA	Feld-programmierbares Gate-Array
GA	Genetischer Algorithmus
gD-MUX	Generalisierter D-MUX
GE	Gatteräquivalent
GNN	Graph Neural Network
GSS	Verallgemeinertes Set-Szenario
HCA	Hill-Climbing Angriff
HD	Hamming-Abstand
HDL	Hardware-Beschreibungssprache
HLS	High-Level Synthese
HT	Hardware-Trojaner
HW	Hardware
IC	Integrierter Schaltkreis
IFS	Identifizieren Sie das Flip-Signal
ILC	Verriegelungsschaltung
IO	Eingabe und Ausgabe
IP	Geistiges Eigentum
ISA	Befehlssatzarchitektur
KBM	Schlüssel-Bit-Zuordnung
KPA	Schlüsselprognosegenauigkeit
LC	Layout-Tarnung
LCALL	Logikkegel-Analyse Logikverriegelung
LCBFA	Logik-Kegel-basierter Brute-Force-Angriff
LCSBLL	Logikkegel-größenbasiertes Logikschloss
LUT	Suchtabelle
LVE	Extraktion von Lokalitätsvektoren
MiG-V	Hergestellt in Deutschland RISC-V
ML	Maschinelles Lernen
MLP	Mehrschichtiges Perzeptron
MUX	Multiplexer
NAS	Neurale Architektursuche
OCP	Optisches kontaktloses Abtasten

OG	Orakel-geleitet
OL	Orakellos
OOP	Objektorientierte Programmierung
PF	Punktfunktion
PSA	Pfad-Sensibilisierungsangriff
PSO	Partikelschwarmoptimierung
RAA	Rationalitätsanalyse-Angriff
RARLL	Redundanzangriffsresistente Logikverriegelung
RE	Reverse Engineering
RLL	Zufällige Logikverriegelung
RNT	Zufälliger Netlist-Test
RSAS	Robustes SAS
RTL	Register-Transfer-Level
SAAM	Strukturanalyseangriff auf MUX-basiertes Locking
SARO	Skalierbare angriffsresistente Verschleierung
SAS	Starkes Anti-SAT
SAT	Boolesches Erfüllbarkeitsproblem
SFLL	Funktionalitätseingeschränkte Logikverriegelung
SGS	Sensibilisierungsgesteuertes SAT
SKRA	Statistischer Schlüssel-Wiederherstellungsangriff
SLL	Starke (sichere) Logikverriegelung
SMT	Erfüllbarkeit Modulo Theorie
SoC	System-auf-Chip
SPS	Signalwahrscheinlichkeitsverzerrung
SRS	Selbstreferenzierendes Szenario
TAAL	Manipulationsangriff auf jeden schlüsselbasierten logisch gesperrten Schaltkreis
TAL	Testbewusstes Sperren
TDM	Test-Daten Mining
TGA	Topologiegeleiteter Angriff
TGARLL	TGA-resistente Logikverriegelung
TRENTOS	Vertrauenswürdiges Entity-Betriebssystem
TRLL	Wirklich zufälliges Logikschloss
TTLock	Hartnäckige und spurlose Logikverriegelung

Notation

$\boldsymbol{C}^f$	($\boldsymbol{C}^f$) Binäres Galois-Feld 2
$\boldsymbol{D}^f$	Funktioneller Täuschungsfaktor
$\boldsymbol{FHS}$	Funktionale Hardware-Sicherheit
G^f	Goldener Faktor
H	Frequenzhistogramm
$\boldsymbol{IC}_{ll}$	Logikgesperrtes Design
$\boldsymbol{IC}$	Original (unverriegeltes) Design
I	Ein einzelnes Eingabemuster
$\boldsymbol{KSS}$	Größe des Schlüsselraums
K	Aktivierungsschlüssel
$\boldsymbol{O}$	Ein einzelnes Ausgabemuster
SHS_{cc}	Strukturelle Komplexitätsänderung
SHS_{en}	Strukturelle Schlüsselgatter-Entropie
SHS	Strukturelle Hardware-Sicherheit
$\boldsymbol{S}^f$	Geheimhaltungsfaktor
$\boldsymbol{TT}$	Wahrheitstabelle
$\mathcal{I}$	Satz aller Eingabemuster
$\mathcal{O}$	Satz aller Ausgabemuster
d_e	Euklidischer Abstand
f^{dc}	Funktion zur Verringerung des Schlüsselraums
f^{kd}	Schlüsselgatter-Verteilungsmaßfunktion
s_d	Strukturverteilungsvektor
s_{max}	Maximaler Strukturabstandsvektor
s_{opt}	Optimaler Strukturverteilungsvektor
v_{ext}	Extrahierter Merkmalsvektor
v_{max}	Maximaler Distanzmerkmalsvektor
v_{opt}	Optimaler Merkmalsvektor

Teil I
Vertrauenswürdige Hardware: Bedrohungen und Lösungen

Kapitel 1
Einführung

Die Computersicherheit ist zu einer treibenden Kraft im Design moderner Elektroniksysteme geworden. Über viele Jahre hinweg wurden Sicherheitsprimitive, insbesondere in der Software, ausgiebig erforscht. Die HW-Sicherheit ist im Vergleich dazu ein relativ junges Feld, da HW traditionell als immun gegen Angriffe angesehen wurde und eine Vertrauensbasis für jedes elektronische System darstellt. In den letzten drei Jahrzehnten wurde jedoch eine zunehmende Anzahl von Schwachstellen identifiziert, deren Ursache in der Hardware selbst liegt [2]. Angriffe, die diese Schwachstellen ausnutzen, können grob in zwei Kategorien unterteilt werden. Die erste Kategorie umfasst alle Angriffsvektoren, die durch einen übersehenen Konstruktionsfehler in der HW-Implementierung ermöglicht werden, was eine Reihe von Angriffsvektoren eröffnet, vorwiegend in Form von Seitenkanalangriffen und Ausnutzung anderer unbeabsichtigter HW-Nebeneffekte. Bemerkenswerte Beispiele in den letzten Jahren sind Transient Execution Attacks, wie Meltdown [11] und Spectre [10], sowie Sicherheitslücken in dynamischen Random-Access-Speichern wie RowHammer [9, 14]. Die zweite Kategorie umfasst neuere Angriffsarten, die durch absichtliche, bösartige Veränderungen in der HW ermöglicht werden, allgemein bekannt als Hardware-Trojaner (HT) [3]. Die durch diese Modifikationen eingeführten Herausforderungen haben tiefgreifende Auswirkungen auf die Forschungs- und Entwicklungslandschaft der Hardware-Sicherheit, insbesondere da sie als Schlüsselaktivatoren einer theoretisch unbegrenzten Angriffsfläche dienen können; einschließlich Informationslecks, Zuverlässigkeitsverschlechterung und Verhinderung der Dienstleistung etc.

Die Einführung von HTs wirft eine interessante Frage auf: Was ist ihre Ursache? Heutzutage haben ein hochkompetitives Umfeld, kurze Time-to-Market-Zeiten und der ständig wachsende Bedarf an reduzierten Design- und Produktionskosten die Integrated Circuit (IC) Supply Chain zu einer globalen Anstrengung gemacht, angetrieben durch Drittanbieter-Intellectual-Property (IP), die Vergabe von Unteraufträgen an externe Designbüros und die Auslagerung der Fertigung an Off-Site-Werke. Diese tiefe Abhängigkeit von externen Teilnehmern hat zu einer

D. Sisejkovic und R. Leupers, *Logic Locking*,
https://doi.org/10.1007/978-3-031-54400-2_1

weitreichenden Konsequenz geführt – dem Verlust von Vertrauen und Sicherheit. Daher stehen legitime IP-Inhaber vor der Möglichkeit, dass HTs injiziert werden, was zu unzuverlässigen HW-Komponenten führt. Und diese Herausforderung ist durchaus eine ernsthafte. Eine breite Palette von Ingenieuren ist an der Gestaltung und Produktion von HW beteiligt, hat vollen Zugriff auf ein Design und arbeitet oft über mehrere Organisationen, Länder und sogar Kontinente hinweg. Es braucht nur eine einzige abtrünnige Einheit, die eine winzige, heimliche und sorgfältig platzierte Modifikation einpflanzt, um den Grundstein für einen katastrophalen Angriff zu legen. Diese verdeckte Natur von HTs macht es schwierig, sie in freier Wildbahn zu identifizieren, insbesondere aufgrund der inhärenten Komplexität moderner Schaltkreise und schrumpfender Größen der Bauteile. Daher wurden nur eine Handvoll mutmaßlicher HTs gemeldet. Zum Beispiel versagte vor mehr als einem Jahrzehnt ein syrisches Radarsystem bei der Warnung vor einem bevorstehenden Luftangriff, angeblich aufgrund von HTs, die in den Verteidigungssystemen eingebettet waren [1, 13]. Obwohl es schwierig ist, die Einbeziehung von HTs in solchen Vorfällen zu verifizieren, ist das bloße Potenzial dieser winzigen, bösartigen Designmodifikation zu einem Schwerpunkt in Forschung und Industrie geworden. Die US-Militär- und Geheimdienstführer haben Hardware-Trojaner zu den schwerwiegendsten Bedrohungen gezählt, denen die Nation im Falle eines Krieges gegenüberstehen könnte [12]. Darüber hinaus hat die US Defense Advanced Research Project Agency (DARPA) mehrere Förderprogramme zur Bewältigung des Problems vertrauenswürdiger Elektronik aufgelegt, darunter das TRUST- [8], IRIS- [6] und SHIELD-Programm [7] u. A. Die Ernsthaftigkeit dieses Problems wurde auch in Deutschland erkannt. Das Bundesministerium für Bildung und Forschung (BMBF) hat ein Rahmenprogramm für 2021–2024 zur Bewältigung der Herausforderungen vertrauenswürdiger und nachhaltiger Mikroelektronik für Deutschland und Europa aufgelegt [4] mit einer Reihe von bereits laufenden Projekten [5].

Die Bemühungen zur Bekämpfung von HT konzentrieren sich auf zwei Schwerpunkte: Erkennung und Prävention. Die Trojaner-Erkennung zielt darauf ab, potenzielle HTs zu erkennen und zu entfernen, möglichst bevor diese in Silizium platziert werden. Allerdings sind Erkennungsansätze noch weit von einer vollständigen Lösung entfernt, aus mehreren Gründen. Erstens können HTs auf vielen verschiedenen Ebenen der HW-Designabstraktion und in verschiedenen Stadien der IC-Lieferkette injiziert werden. Dies macht es schwierig, einen effektiven Erkennungsmechanismus abzuleiten. Zweitens, selbst wenn umfassende (und oft zerstörerische) Reverse-Engineering-Verfahren eingesetzt werden, um das Fehlen von Trojanern in Chips nach der Produktion zu verifizieren, garantiert dies nicht, dass alle produzierten integrierten Schaltkreise (IC) HT-frei sind. Daher wurde mehr Augenmerk auf die Verhinderung der HT-Einführung durch das Design gelegt. Insbesondere hat sich Logic Locking als führende Technik zur Abwehr von HT-Einführungen durch schlüsselgesteuerte funktionale und strukturelle Designänderungen entwickelt, die darauf abzielen, das Asset – das HW-Design – während der gesamten IC-Lieferkette zu schützen [18]. Dabei basiert der Abwehrmechanismus auf der Annahme, dass ein Angreifer umfangreiches

Reverse Engineering durchführen muss, um einen verständlichen, designspezifischen Hardware-Trojaner einzufügen und zu konstruieren. Daher erhöhen die durch die Verriegelung induzierten Änderungen die Komplexität des Angriffs, indem sie die korrekten Verhaltens- und Strukturmerkmale der HW an einen geheimen Schlüssel binden. Dennoch war die evolutionäre Zeitleiste des Logic Lockings mit einer Vielzahl von Angriffsvektoren und unklaren Sicherheitszielen durchsetzt. Dies hat dazu geführt, dass Logic Locking weitgehend ein theoretisches Konzept ohne greifbares Ergebnis geblieben ist.

Um dieses Problem zu lösen, zielen wir in diesem Buch darauf ab, die Praxislücke im Logic Locking zu schließen, indem wir eine Reihe von Modellen, Softwaretools, Angriffen und Verschlüsselungen entwickeln, die die Bewertung und Anwendung des Logic Locking auf komplexe, siliziumbewährte HW-Designs innerhalb eines präzisen und realistischen Angriffsszenarios ermöglichen [15].

1.1 Übersicht

Dieses Buch ist in vier Teile gegliedert, die elf Kapitel abdecken. Die Struktur soll den Leser von grundlegenden Konzepten zur Hardware-Sicherheit bis hin zu Software-Implementierungsdetails für siliziumbereites Logic Locking führen. Am Ende des Buches sollten die Leser in der Lage sein zu verstehen, wie Logic Locking funktioniert und wie es herausgefordert werden kann, wie man die richtigen Werkzeuge zur Implementierung von Verschlüsselungsalgortihmen einsetzt und schließlich, wie man die Sicherheit von Logic Locking mit aufkommenden maschinenlernbasierten Ansätzen bewertet. Das Buch ist wie folgt strukturiert.

Hintergrund Zunächst werden Vorläufer zu Bedrohungen und Lösungen in der Elektroniklieferkette in Kap. 2 vorgestellt.

Hardware-Trojaner Kap. 3 führt die Anatomie von Hardware-Trojanern zusammen mit bestehenden Klassifikationssystemen ein. Darüber hinaus wird eine konsolidierte Klassifikation eingeführt, die die Auswirkungen von Abwehransätzen berücksichtigt. Schließlich vergleicht das Kapitel die Wirksamkeit bestehender Gegenmaßnahmen zur Trojaner-Einführung hinsichtlich der erstellten Klassifikation.

Arbeitsprinzipien und Angriffsszenarien Die Mechanik des Logic Lockings, seine Auswirkungen auf das Reverse Engineering sowie gängige Angriffsszenarien werden in Kap. 4 diskutiert.

Angriffe und Schemata Eine Übersicht und Klassifikation von Entschärfungsangriffen und Logic-Locking-Verfahren wird in Kap. 5 präsentiert.

Sicherheitsmetriken Kap. 6 führt das Design einer der ersten allgemeinen Hardware-Sicherheitsmetriken in Bezug auf das Logic Locking ein. Darüber hinaus wird auf der Grundlage der eingeführten Konzepte das Problem der Sicherheits-Kosten-Abwägung

durch eine Fallstudie analysiert, die die Auswirkungen eines höheren Kostenbudgets auf die Sicherheitseigenschaften des Logic Lockings bewertet.

Software-Framework Das Design und die Implementierung eines softwarebasierten Logic-Locking-Frameworks zum Schutz komplexer Multi-Modul-HW-Designs wird in Kap. 7 diskutiert. Das Framework ist in Form eines End-to-End-Verriegelungsverfahrens konzipiert, das eine technologieunabhängige Designrepräsentation und eine erweiterbare Codebasis für schnelles Verschlüsselung-Prototyping bietet. Darüber hinaus stellt das erstellte Framework sicher, dass Logic Locking in einer industriereifen Umgebung ohne Beeinträchtigung der traditionellen Design-, Verifizierungs- und Fertigungsschritte eingesetzt wird.

Schutz der Prozessorintegrität Die Implementierung von Framework-Erweiterungen in Form von zwei Schutzverfahren, Inter-Lock und Control-Lock, wird in Kap. 8 vorgestellt. Inter-Lock verkörpert ein modulübergreifendes, logikverriegelndes Meta-Verfahren, das jede Verschlüsselung über mehrere HW-Module skaliert und dadurch zusätzliche funktionale und strukturelle Abhängigkeiten zwischen den ausgewählten Komponenten schafft. Dieses modulübergreifende Verfahren ist das erste, welches Sicherheitsimplikationen des Logic Lockings auf komplexe Hardware-Designs erweitert. Control-Lock implementiert einen intermodularen Verschlüsselungsmechanismus, der darauf abzielt, kritische HW-Steuerungssignale gegen die Ausnutzung durch softwaregesteuerte Hardware-Trojaner zu schützen. Die Auswirkungen beider Verfahren werden an siliziumbewährten RISC-V-Prozessorkernen bewertet. Schließlich werden die vorgestellten Forschungsenergebnisse erfolgreich in die Industrie übertragen, was in dem ersten umfassend logikverriegelten und kommerziell erhältlichen Prozessor resultiert – dem "Made in Germany RISC-V"- (MiG-V-)Kern [17]. Damit wird ein wichtiger Meilenstein im Bereich des Logic Locking erreicht.

Sicherheitsbewertung mit maschinellem Lernen Die Einführung grundlegender Konzepte in Angriffe und Abwehrmaßnahmen im Logic Locking in Bezug auf maschinelles Lernen (ML) [16] wird in Kap. 9 vorgestellt. Zunächst wird SnapShot vorgestellt; ein Angriff, der künstliche neuronale Netzwerke nutzt, um korrekte Schlüsselbits direkt aus einer verriegelten Netzliste vorherzusagen. Darüber hinaus wird ein neuroevolutionäres Verfahren entwickelt, um automatisch geeignete neuronale Architekturen für das ausgewählte Vorhersageproblem zusammenzustellen. Darüber hinaus werden das generalisierte Set und das selbstreferenzierende Angriffsszenario als Standardangriffsvektoren in einer maschinenlernbasierten Umgebung diskutiert.

Entwurf eines täuschenden Logic Locking Auf der Grundlage der in Kap. 9 gewonnenen Erkenntnisse wird in Kap. 10 der erste theoretische Test zur Aufdeckung von strukturellen Leckstellen eingeführt. Der Test verkörpert ein Verfahren, das zur Identifizierung grundlegender Sicherheitslücken führen kann, die durch maschinelles Lernen (ML-)getriebene Angriffe ausnutzbar sind. Die Analyseergebnisse dienen als Grundlage für den Aufbau einer Multiplexer-basierten

Verriegelungstechnik, die auf Lernresistenz abzielt. Durch weitere Evaluierungsschritte wird eine Analyse der Herausforderungen bei der ML-resistenten Verriegelung durchgeführt. Darüber hinaus wird ein neuartiger Angriff vorgestellt, der einen großen Irrtum in bestehenden Multiplexer-basierten Logikverschlüsselungen aufdeckt. Die eingeführten Konzepte, Richtlinien und Angriffe bieten das Potenzial, die Grundsteine für das Design der nächsten Generation von Logic Locking im Zeitalter des maschinellen Lernens zu legen.

Nächste Schritte Neue Forschungsrichtungen und offene Herausforderungen werden in Kap. 11 diskutiert. Schließlich schließt Kap. 12 das Buch ab.

Literatur

1. S. Adee, The hunt for the kill switch. IEEE Spectr. **45**(5), 34–39 (2008). https://doi.org/10.1109/MSPEC.2008.4505310
2. S. Bhunia, M. Tehranipoor, *Hardware security: a hands-on learning approach*, 1. Aufl. (Morgan Kaufmann Publishers, San Francisco, 2018)
3. S. Bhunia, M. Tehranipoor, *The hardware trojan war: attacks, myths, and defenses* (Springer, 2018). https://doi.org/10.1007/978-3-319-68511-3
4. Bundesministerium für Bildung und Forschung (BMBF), Mikroelektronik. Vertrauenswürdig und nachhaltig. Für Deutschland und Europa. Rahmenprogramm der Bundesregierung für Forschung und Innovation 2021–2024 (2021). https://www.elektronikforschung.de/rahmenprogramm
5. Bundesministerium für Bildung und Forschung (BMBF), Vertrauenswürdige Elektronik. Forschung und Innovation für technologische Souveränität (2021). https://www.elektronikforschung.de/service/publikationen/vertrauenswuerdige-elektronik
6. Defense Advanced Research Project Agency (DARP), Integrity and reliability of integrated circuits (IRIS) (2021). https://www.darpa.mil/program/integrity-and-reliability-of-integrated-circuits
7. Defense Advanced Research Project Agency (DARP), Supply chain hardware integrity for electronics defense (SHIELD) (2021). https://www.darpa.mil/program/supply-chain-hardware-integrity-for-electronics-defense
8. Defense Advanced Research Project Agency (DARP), Trusted integrated circuits (TRUST) (2021). https://www.darpa.mil/program/trusted-integrated-circuits
9. Y. Kim, R. Daly, J. Kim, C. Fallin, J.H. Lee, D. Lee, C. Wilkerson, K. Lai, O. Mutlu, Flipping bits in memory without accessing them: an experimental study of dram disturbance errors, in 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA) (2014), S. 361–372. https://doi.org/10.1109/ISCA.2014.6853210
10. P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, Y. Yarom, Spectre attacks: exploiting speculative execution, in *2019 IEEE Symposium on Security and Privacy (SP)* (2019), S. 1–19. https://doi.org/10.1109/SP.2019.00002
11. M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, M. Hamburg, Meltdown: reading kernel memory from user space, in *27th USENIX Security Symposium (USENIX Security 18)* (USENIX Association, Baltimore, 2018), S. 973–990. https://www.usenix.org/conference/usenixsecurity18/presentation/lipp
12. J. Markoff, Old trick threatens the newest weapons. The New York Times nytimes.com (2009). www.nytimes.com/2009/10/27/science/27trojan.html
13. S. Mitra, H.S.P. Wong, S. Wong, The Trojan-proof chip. IEEE Spectr. **52**(2), 46–51 (2015). https://doi.org/10.1109/MSPEC.2015.7024511

14. O. Mutlu, J.S. Kim, RowHammer: a retrospective. Trans. Comput.-Aided Des. Integr. Circuits Syst. **39**(8), 1555–1571 (2020). https://doi.org/10.1109/TCAD.2019.2915318
15. D. Sisejkovic, Designing trustworthy hardware with logic locking. Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, Aachen (2022). https://doi.org/10.18154/RWTH-2022-02625
16. D. Sisejkovic, L.M. Reimann, E. Moussavi, F. Merchant, R. Leupers, Logic locking at the frontiers of machine learning: a survey on developments and opportunities, in *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)* (2021), S. 1–6. https://doi.org/10.1109/VLSI-SoC53125.2021.9606979
17. D. Sisejkovic, F. Merchant, L.M. Reimann, R. Leupers, M. Giacometti, S. Kegreiß, A secure hardware-software solution based on RISC-V, logic locking and microkernel, in *Proceedings of the 23th International Workshop on Software and Compilers for Embedded Systems, SCOPES '20* (Association for Computing Machinery, New York, 2020), S. 62–65. https://doi.org/10.1145/3378678.3391886
18. M. Yasin, J.J. Rajendran, O. Sinanoglu, *Trustworthy hardware design: combinational logic locking techniques* (Springer, Berlim, 2020). https://doi.org/10.1007/978-3-030-15334-2

Kapitel 2
Hintergrundinformationen

Um den Inhalt dieses Buches besser zu verstehen, führt dieses Kapitel die folgenden Grundlagen ein. Ein Überblick über die wichtigsten Sicherheitslücken in der elektronischen Lieferkette wird in Abschn. 2.1 vorgestellt. Die wichtigsten Design-for-Trust-(DfTr-)Lösungen werden in Abschn. 2.2 detailliert beschrieben. Abschn. 2.3 schließt das Kapitel ab.

2.1 Gefahren in der Elektronik-Lieferkette

Die Komplexität und dezentrale Natur der modernen Elektronik-Lieferkette hat zu einem Mangel an Vertrauen und Sicherheit geführt. Folglich wurde eine Reihe von Angriffsvektoren eingeführt, um integrierte Schaltkreise (IC) zu stehlen, illegitim zu verkaufen oder deren Integrität zu kompromittieren. Die folgenden Unterabschnitte stellen den Hintergrund zu ausgewählten Vertrauensproblemen dar. Weitere Details finden Sie in [2].

2.1.1 Reverse Engineering

Im Kontext von Hardware wird Reverse Engineering (RE) als der Prozess definiert, bei dem jemand, der nicht der ursprüngliche Designbesitzer ist, einen Satz von Spezifikationen für ein Hardware-Design extrahiert [32]. Dabei kann RE auf verschiedenen Schaltungsebenen eingesetzt werden [28]. Die Legitimität von RE hängt davon ab, wofür das Ergebnis verwendet wird. Daher kann das Produkt von RE entweder für Verifizierungszwecke oder illegale Handlungen wie die Einführung von Hardware-Trojanern oder den Diebstahl von geistigem Eigentum (IP) genutzt werden.

D. Sisejkovic und R. Leupers, *Logic Locking*,
https://doi.org/10.1007/978-3-031-54400-2_2

Der Prozess des Reverse Engineering von Hardware-Designs umfasst eine Reihe von manuellen und halbautomatisierten Schritten [3, 15, 39, 41, 43]. Ausgehend von einem gefertigten IC kann der RE-Fluss in Netlist-Extraktion und Funktionsidentifikation unterteilt werden. Ersteres extrahiert eine Netlist-Darstellung des physischen Chips durch mehrere aufeinanderfolgende Schritte, einschließlich der Probenvorbereitung (Paketentfernung und Entschichtung), Bildaufnahme, Layout-Extraktion und Netlist-Generierung. Letzteres betrifft die Erfassung einer High-Level-Beschreibung der beabsichtigten Funktionalität des Designs [5].

Aufgrund seiner Komplexität hat der Prozess des Reverse Engineering keine klaren Richtlinien. Erst kürzlich wurden erste Versuche unternommen, die erforderlichen kognitiven und technischen Fähigkeiten zur Durchführung von RE zu analysieren [6]. Dennoch bleibt es eine Herausforderung, den Prozess vollständig zu automatisieren sowie die Komplexität von RE für ein spezifisches Design zu quantifizieren.

2.1.2 Hardware-Trojaner

Hardware-Trojaner (HT) sind als bösartige und absichtliche Schaltungsmodifikationen definiert, die nach der Implementierung zu unerwünschtem Schaltungsverhalten führen können [9, 44]. Das bösartige Verhalten kann sich in Form von Informationslecks, Leistungsabfall, erhöhter Leistungsaufnahme, Denial-of-Service-(DoS-)Angriffen und anderem manifestieren.

Die Anatomie von Hardware-Trojanern besteht aus einem Auslöser und einem Payload (Schadwirkgung) [7, 10]. Der Auslöser aktiviert den Trojaner basierend auf einem spezifischen Aktivierungsereignis, wie dem Auftreten spezifischer Datenwerte oder Schaltungszustände, externen Signalen, Anzahl von Zyklen und anderen. Das bösartige Verhalten des Trojaners manifestiert sich in Form des Payloads. Die bösartige Schaltung kann auf verschiedenen Designebenen in die Hardware eingefügt werden, abhängig davon, welche Einheiten in der Lieferkette als vertrauenswürdig angesehen werden. Daher können HT grundsätzlich während der Spezifikation, des Designs, der Fertigung, der Prüfung oder der Montage und Verpackung in ein Design eingeführt werden, wobei sie von nicht vertrauenswürdigem Personal oder Electronic-Design-Automation-(EDA-)Tools initiiert werden. Diese breite Angriffslandschaft hat zur Einführung vieler Hardware-Trojaner-Taxonomien und Beispielimplementierungen geführt [13, 21, 34, 36, 40, 47]. Darüber hinaus hat eine aktuelle Studie sogar gezeigt, wie Hardware-Trojaner mühelos und automatisch in fertige Layouts implantiert werden können [27].

Die vielfältigen Designmöglichkeiten, Einfügungsorte und die heimliche Implementierungsnatur machen die Erkennung von HTs zu einer herausfordernden Aufgabe. Darüber hinaus wird es, ähnlich wie bei regulären Fehlern, je später Trojaner im Design- und Produktionsfluss erkannt werden, umso

kostspieliger und schwieriger für den IP-Besitzer zu handeln. Dieses Problem wird noch dadurch verschärft, dass oft angenommen wird, dass HTs von nicht vertrauenswürdigen, externen Fertigungsanlagen eingesetzt werden – außerhalb der Kontrolle des legitimen IP-Besitzers.

Ein weiterer wichtiger Aspekt liegt in den Ressourcen, die für das Design, die Implementierung und die Injektion von Hardware-Trojanern erforderlich sind [19, 38]. Im Allgemeinen können *design-unabhängige* HT mit sehr wenig Wissen über die Funktionalität oder Struktur des Designs implementiert und eingefügt werden. Diese Trojaner werden jedoch wahrscheinlich erkannt, zeigen unkontrollierbare Auslösemechanismen und führen zu zufälligen Payloads, die einem zufälligen Fehler ähneln. Im Gegensatz dazu können *design-abhängige* Trojaner so konstruiert werden, dass sie eine kontrollierbare Aktivierung, heimliche Implementierung und spezifische Payloads ermöglichen, was zu hochwirksamen Angriffsszenarien führt. Folglich erfordern designabhängige HTs, dass der Angreifer das Design rekonstruiert (durch RE) und die Funktionalität und Struktur des Designs versteht, um die bösartigen Modifikationen erfolgreich zu konstruieren und einzufügen [49, 50]. Weitere Details zu Hardware-Trojanern werden in Kap. 3 vorgestellt.

2.1.3 IP-Piraterie und -Übernutzung

IP-Piraterie bezieht sich auf die illegale oder unlizenzierte Nutzung von geistigem Eigentum [49]. Der Akt der Piraterie kann an verschiedenen Punkten in der elektronischen Lieferkette auftreten. Beispielsweise könnte ein unehrlicher System-on-Chip-(SoC-)Designer illegitime Kopien eines legal erworbenen geistiges Eigentum von Dritten (3PIP; Third-Party Intellectual Property) anfertigen, um sie an andere SoC-Designer zu verkaufen. Darüber hinaus kann der SoC-Designer bestimmte Modifikationen vornehmen und das veränderte Design als neues IP verkaufen. Piraterie kann auch in späteren Design- und Fertigungsstadien auftreten. Beispielsweise können böswillige Akteure illegale Kopien des IP (im Netlist- oder Layout-Format) verkaufen, wenn ein IP-Design an externe Designhäuser oder Fertigungsanlagen ausgelagert wird [8].

Eine weitere Bedrohung ist als IP-Übernutzung bekannt, bei der eine Entität das IP in mehr Instanzen verwendet als vertraglich vereinbart [2]. Im Allgemeinen gehört das IP rechtlich dem IP-Autor. SoC-Integratoren verlassen sich oft darauf, IP von legitimen IP-Besitzern zu kaufen, um einen Chip zu entwerfen. Daher können IP-Besitzer die Integration ihrer IP an den SoC-Designer für eine bestimmte Anzahl von Chips lizenzieren. Ein unehrlicher SoC-Designer könnte jedoch das IP übernutzen, indem er die Produktion von mehr Chips als im vereinbarten Lizenzschema vorgesehen anordnet, was zu Verlusten für den IP-Besitzer führt. Es ist zu beachten, dass der Halbleiter-IP-Markt im Jahr 2019 auf US 5,33 Mrd geschätzt wurde und voraussichtlich bis 2027 US 7,44 Mrd erreichen wird [42]. Daher haben echte IP-Besitzer einen starken wirtschaftlichen Anreiz, ihre Vermögenswerte zu schützen.

2.1.4 IC-Überbau und Fälschung

Unzuverlässige Fertigungsanlagen und Montageeinrichtungen können mehr Chips produzieren als vertraglich vereinbart, was zu einem IC-Überbau führt. Dieses Vertrauensproblem entsteht aus der Unfähigkeit des IP-Eigentümers und des Designhauses, den Fertigungs- und Montageprozess zu überwachen. Die bösartigen Einheiten können Chips mit hohem Gewinn überbauen, da keine Designentwicklungskosten erforderlich sind. Darüber hinaus kann der Überbau praktisch ohne Kosten durchgeführt werden, indem eine geringere Ausbeute gemeldet wird [8, 33].

In diesem Zusammenhang stellt die IC-Fälschung eine ernsthafte Bedrohung dar. Gefälschte ICs sind Nachbildungen der echten integrierten Schaltung, die funktional und visuell identisch mit den echten IC erscheinen. Dazu gehören umetikettierte, außerhalb der Spezifikation liegende oder recycelte ICs, die oft aus ausrangierten elektronischen Geräten extrahiert werden [49]. Neben den Umsatzeinbußen für legitime IP-Eigentümer sind Überbau und gefälschte Komponenten eine Hauptquelle für ernsthafte Zuverlässigkeits- und Sicherheitsprobleme. Mit minimaler oder keiner Prüfung und Verifizierung haben diese IC-Auswirkungen auf Personalcomputer, Telekommunikation, Automobil-Elektronik, medizinische Geräte und militärische Systeme [16, 17]. Zum Beispiel hat das Pentagon – das höchste militärische Hauptquartier der Vereinigten Staaten – berichtet, dass etwa 15 % aller gekauften elektronischen Teile gefälscht sind. Darüber hinaus wird geschätzt, dass die Fälschung von elektronischen Teilen einen jährlichen Verlust von mehr als US$ 7,5 Mrd für legitime Halbleiterhersteller verursacht [35].

2.2 Design-for-Trust-Lösungen

In den letzten Jahrzehnten wurde eine Reihe von Design-for-Trust-(DfTr-)Lösungen vorgeschlagen, um Sicherheit und Vertrauen in der gesamten IC-Lieferkette durchzusetzen. Im Folgenden stellen wir einen Teil dieser Lösungen vor, die sich auf den *Schutz der Hardware-Integrität* konzentrieren.

2.2.1 Layout-Tarnung

Die Layout-Tarnung (LC; Layout Camouflaging) zielt darauf ab, das Reverse Engineering zu verhindern, indem ausgewählte Zellen durch ihre getarnten Gegenstücke ersetzt werden [29]. Die getarnten Zellen haben eine strukturell ähnliche Konstruktion über verschiedene Gattungstypen hinweg. Aus der Sicht des RE muss der Angreifer daher weitere Anstrengungen unternehmen, um die korrekte Funktionalität aufzudecken. LC kann durch Verschleierung der Verbindungen