

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                   |                                |                                                         |                                                          |                                                 |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|--------------------------------|---------------------------------------------------------|----------------------------------------------------------|-------------------------------------------------|
| <b>REPORT DOCUMENTATION PAGE</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                   |                                | Form Approved OMB NO. 0704-0188                         |                                                          |                                                 |
| <p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA, 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p> |                   |                                |                                                         |                                                          |                                                 |
| 1. REPORT DATE (DD-MM-YYYY)<br>14-12-2017                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                   | 2. REPORT TYPE<br>Final Report |                                                         | 3. DATES COVERED (From - To)<br>1-Aug-2013 - 31-Jul-2015 |                                                 |
| 4. TITLE AND SUBTITLE<br>Final Report: Workshop on Trustworthy Hardware - NYU-Poly                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                   |                                | 5a. CONTRACT NUMBER<br>W911NF-13-1-0272                 |                                                          |                                                 |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                   |                                | 5b. GRANT NUMBER                                        |                                                          |                                                 |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                   |                                | 5c. PROGRAM ELEMENT NUMBER                              |                                                          |                                                 |
| 6. AUTHORS                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                   |                                | 5d. PROJECT NUMBER                                      |                                                          |                                                 |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                   |                                | 5e. TASK NUMBER                                         |                                                          |                                                 |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                   |                                | 5f. WORK UNIT NUMBER                                    |                                                          |                                                 |
| 7. PERFORMING ORGANIZATION NAMES AND ADDRESSES<br>Polytechnic University - Brooklyn<br>Office of Sponsored Research<br>Polytechnic Institute of New York University<br>Brooklyn, NY 11201 -3840                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                   |                                | 8. PERFORMING ORGANIZATION REPORT NUMBER                |                                                          |                                                 |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)<br>U.S. Army Research Office<br>P.O. Box 12211<br>Research Triangle Park, NC 27709-2211                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                   |                                | 10. SPONSOR/MONITOR'S ACRONYM(S)<br>ARO                 |                                                          |                                                 |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                   |                                | 11. SPONSOR/MONITOR'S REPORT NUMBER(S)<br>64123-CS-CF.6 |                                                          |                                                 |
| 12. DISTRIBUTION AVAILABILITY STATEMENT<br>Approved for public release; distribution is unlimited.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                   |                                |                                                         |                                                          |                                                 |
| 13. SUPPLEMENTARY NOTES<br>The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                   |                                |                                                         |                                                          |                                                 |
| 14. ABSTRACT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                   |                                |                                                         |                                                          |                                                 |
| 15. SUBJECT TERMS                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                   |                                |                                                         |                                                          |                                                 |
| 16. SECURITY CLASSIFICATION OF:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                   |                                | 17. LIMITATION OF ABSTRACT<br>UU                        | 15. NUMBER OF PAGES                                      | 19a. NAME OF RESPONSIBLE PERSON<br>Ramesh Karri |
| a. REPORT<br>UU                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | b. ABSTRACT<br>UU | c. THIS PAGE<br>UU             |                                                         |                                                          | 19b. TELEPHONE NUMBER<br>646-997-3596           |

**RPPR Final Report**  
as of 14-Mar-2018

Agency Code:

Proposal Number: 64123CSCF  
**INVESTIGATOR(S):**

**Agreement Number: W911NF-13-1-0272**

**Name:** PhD Ramesh Karri  
**Email:** rkarri@nyu.edu  
**Phone Number:** 6469973596  
**Principal:** Y

Organization: **Polytechnic University - Brooklyn**

Address: Office of Sponsored Research, Brooklyn, NY 112013840

Country: USA

DUNS Number: 066076225

EIN: 111630820

**Report Date:** 31-Oct-2015

Date Received: 14-Dec-2017

**Final Report** for Period Beginning 01-Aug-2013 and Ending 31-Jul-2015

**Title:** Workshop on Trustworthy Hardware - NYU-Poly

**Begin Performance Period:** 01-Aug-2013

**End Performance Period:** 31-Jul-2015

**Report Term:** 0-Other

Submitted By: PhD Ramesh Karri

Email: rkarri@nyu.edu

Phone: (646) 997-3596

**Distribution Statement:** 1-Approved for public release; distribution is unlimited.

**STEM Degrees:** 2

**STEM Participants:**

**Major Goals:** The first major goal was to organize a workshop Hardware security and trustworthy hardware. The second major goal is to create an open-access tutorial style edited book in trustworthy hardware

**Accomplishments:** 1. Organized workshop on Trustworthy Hardware  
2. Published a Proceedings of IEEE Issue on Trustworthy Hardware with 10 tutorial articles

**Training Opportunities:** The issue is indirectly used either as a reference or as a monograph in hardware security classes.

**Results Dissemination:** Proceeding of IEEE Special Issue on Trustworthy Hardware (August 2014 Issue, please see products page) has come out. It contains 10 open-access, tutorial style articles. The issue is widely referenced and cited.

**Honors and Awards:** Papers in the Proceedings of the IEEE open access issue Aug 2014 are highly cited and referenced.

**Protocol Activity Status:**

**Technology Transfer:** Nothing to Report

**ARTICLES:**

## RPPR Final Report as of 14-Mar-2018

**Publication Type:** Journal Article      Peer Reviewed: N      **Publication Status:** 1-Published  
**Journal:** Proceedings of the IEEE  
**Publication Identifier Type:** DOI      **Publication Identifier:** 10.1109/JPROC.2014.2334837  
**Volume:** 102      **Issue:** 8      **First Page #:** 1123  
**Date Submitted:** 11/30/17 12:00AM      **Date Published:** 8/1/14 12:00PM  
**Publication Location:** IEEE, USA  
**Article Title:** Trustworthy Hardware [Scanning the Issue]  
**Authors:** Ramesh Karri, Farinaz Koushanfar  
**Keywords:** Trustworthy Hardware  
**Abstract:** Systems should be made trustworthy and secure hardware up. Security and trust should be considered as a first class metric during all stages in the IC and system design flows side-by-side power, cost, and performance. This special issue showcases advances in state-of-the-art trustworthy ICs and systems, and spans several aspects of IC and system security ranging from theoretical and conceptual foundations, synthesis, testing, and verification, through modeling and optimization, to case studies. From a trustworthy hardware platforms perspective this special issue covers IC and system security and trust issues in application-specific integrated circuits (ASICs), custom off the shelf (COTS), field-programmable gate arrays (FPGAs), microprocessors, and embedded systems. It explains novel hardware security primitives such as physical unclonable functions (PUFs), and public PUFs (PPUFs), among others.  
**Distribution Statement:** 1-Approved for public release; distribution is unlimited.  
**Acknowledged Federal Support:** Y

**Publication Type:** Journal Article      Peer Reviewed: Y      **Publication Status:** 1-Published  
**Journal:** Proceedings of the IEEE  
**Publication Identifier Type:** DOI      **Publication Identifier:** 10.1109/JPROC.2014.2332154  
**Volume:** 102      **Issue:** 8      **First Page #:** 1266  
**Date Submitted:** 11/30/17 12:00AM      **Date Published:** 8/1/14 4:00AM  
**Publication Location:** IEE, USA  
**Article Title:** Regaining Trust in VLSI Design: Design-for-Trust Techniques  
**Authors:** Jeyavijayan Rajendran, Ozgur Sinanoglu, Ramesh Karri  
**Keywords:** VLSI Trust, Hardware Security, Test  
**Abstract:** Designers use third-party intellectual property (IP) cores and outsource various steps in their integrated circuit (IC) design flow, including fabrication. As a result, security vulnerabilities have been emerging, forcing IC designers and end-users to reevaluate their trust in hardware. If an attacker gets hold of an unprotected design, attacks such as reverse engineering, insertion of malicious circuits, and IP piracy are possible. In this paper, we shed light on the vulnerabilities in very large scale integration (VLSI) design and fabrication flow, and survey design-for-trust (DfTr) techniques that aim at regaining trust in IC design. We elaborate on four DfTr techniques: logic encryption, split manufacturing, IC camouflaging, and Trojan activation. These techniques have been developed by reusing VLSI test principles.  
**Distribution Statement:** 1-Approved for public release; distribution is unlimited.  
**Acknowledged Federal Support:** Y

## RPPR Final Report as of 14-Mar-2018

**Publication Type:** Journal Article      Peer Reviewed: Y      **Publication Status:** 1-Published  
**Journal:** Proceedings of the IEEE  
**Publication Identifier Type:** DOI      **Publication Identifier:** 10.1109/JPROC.2014.2335155  
**Volume:** 102      **Issue:** 8      **First Page #:** 1283  
**Date Submitted:** 11/30/17 12:00AM      **Date Published:** 8/1/14 4:00AM  
**Publication Location:** IEEE, USA  
**Article Title:** A Primer on Hardware Security: Models, Methods, and Metrics  
**Authors:** Masoud Rostami, Farinaz Koushanfar, Ramesh Karri  
**Keywords:** Hardware Security, Security Metrics  
**Abstract:** The multinational, distributed, and multistep nature of integrated circuit (IC) production supply chain has introduced hardware-based vulnerabilities. Existing literature in hardware security assumes ad hoc threat models, defenses, and metrics for evaluation, making it difficult to analyze and compare alternate solutions. This paper systematizes the current knowledge in this emerging field, including a classification of threat models, state-of-the-art defenses, and evaluation metrics for important hardware-based attacks.  
**Distribution Statement:** 1-Approved for public release; distribution is unlimited.  
**Acknowledged Federal Support:** Y

### BOOKS:

**Publication Type:** Book      Peer Reviewed: Y      **Publication Status:** 1-Published  
**Publication Identifier Type:** Other      **Publication Identifier:** Proceedings of IEEE  
**Book Edition:** 1      **Volume:** 108      **Publication Year:** 2014      **Date Received:** 30-Nov-2017  
**Publication Location:** USA (August Issue)  
**Publisher:** Proceedings of the IEEE  
**Book Title:** Special Issue on Trustworthy Hardware (Open access), Proceedings of the IEEE  
**Authors:** Ramesh Karri, Farinaz Koushanfar  
**Editor:**  
**Acknowledged Federal Support:** Y

### WEBSITES:

**URL:** <http://isis.poly.edu/~jv/aro/index.html>  
**Date Received:** 30-Nov-2017  
**Title:** Workshop on Trustworthy Hardware  
**Description:** Hardware security and trustworthy hardware are on the brink of a revolution and this workshop and the resulting special issue/book seek to showcase the latest advances in state-of-the-art trustworthy hardware and identify the most promising research frontiers in IC and system security. The scope of this workshop spans critical aspects of IC and system security ranging from theoretical and conceptual foundations, synthesis, testing and verification, to modeling and optimization to case studies. From a platforms perspective this workshop/special issue will cover IC and system security and trust issues in ASICs, COTS, FPGAs, microprocessors/DSPs, and embedded systems.

August 2014 | Volume 102 | Number 8

# Proceedings OF THE IEEE

SPECIAL ISSUE

## Trustworthy Hardware

Point of View: Energy Return  
On Energy Invested

Scanning Our Past:  
STARS: Software Industry





## SPECIAL ISSUE

### TRUSTWORTHY HARDWARE

Edited by R. Karri and F. Koushanfar

- 1126 Physical Unclonable Functions and Applications: A Tutorial**  
By C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas  
[INVITED PAPER] This paper is a tutorial on ongoing work in physical-disorder-based security, security analysis, and implementation choices.
- 1142 Public Physical Unclonable Functions**  
By M. Potkonjak and V. Goudar  
[INVITED PAPER] This paper surveys the time-bounded or public PUFs, including their design and security evaluation and the new protocols that they can support.
- 1157 Microcontrollers as (In)Security Devices for Pervasive Computing Applications**  
By D. Strobel, D. Oswald, B. Richter, F. Schellenberg, and C. Paar  
[INVITED PAPER] This paper discusses possible threats to embedded systems using two microcontroller case studies.
- 1174 Trustworthiness of Medical Devices and Body Area Networks**  
By M. Zhang, A. Raghunathan, and N. K. Jha  
[CONTRIBUTED PAPER] This paper surveys the threat landscape of medical embedded devices and the merits and shortcomings of existing defenses.
- 1189 Mobile Trusted Computing**  
By N. Asokan, J.-E. Ekberg, K. Kostianen, A. Rajan, C. Rozas, A.-R. Sadeghi, S. Schulz, and C. Wachsmann  
[INVITED PAPER] This paper surveys the trusted computing features in mobile computing platforms.
- 1207 Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain**  
By U. Guin, K. Huang, D. DiMase, J. M. Carulli, Jr., M. Tehranipoor, and Y. Makris  
[INVITED PAPER] This paper surveys the state of the art in counterfeiting and detection technologies.
- 1229 Hardware Trojan Attacks: Threat Analysis and Countermeasures**  
By S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan  
[INVITED PAPER] This paper is a survey of the state-of-the-art Trojan attacks, modeling, and countermeasures.
- 1248 FPGA Security: Motivations, Features, and Applications**  
By S. M. Trimberger and J. J. Moore  
[INVITED PAPER] This paper discusses all aspects of FPGA security and trust.
- 1266 Regaining Trust in VLSI Design: Design-for-Trust Techniques**  
By J. Rajendran, O. Sinanoglu, and R. Karri  
[INVITED PAPER] This paper surveys how concepts in VLSI test can be adopted in the context of hardware security and trust and design-for-trust techniques.
- 1283 A Primer on Hardware Security: Models, Methods, and Metrics**  
By M. Rostami, F. Koushanfar, and R. Karri  
[INVITED PAPER] The paper is a primer on hardware security threat models, metrics, and remedies.

## DEPARTMENTS

- 1118 POINT OF VIEW**  
Energy Return On Energy Invested (EROI): A Quintessential but Possibly Inadequate Metric for Sustainability in a Solar-Powered World?  
By W. F. Pickard
- 1123 SCANNING THE ISSUE**  
Trustworthy Hardware  
By R. Karri and F. Koushanfar
- 1296 SCANNING OUR PAST**  
STARS: Software Industry  
By B. Grad
- 1304 FUTURE SPECIAL ISSUES/SPECIAL SECTIONS**



**On the Cover:** On this month's cover, security issues are highlighted in general by the illustration of the Trojan horse which the Greeks used successfully to breach the city of Troy.



# Trustworthy Hardware

By RAMESH KARRI

Guest Editor

FARINAZ KOUSHANFAR

Guest Editor

Since the 1990s, there has been a steady trend away from inhouse integrated circuit (IC) design and fabrication toward outsourcing various aspects of design, fabrication, testing, and packaging of ICs. The emergence of such a globalized, horizontal semiconductor business model created hitherto unknown security and trust concerns in the ICs and the information systems (rooted in these ICs), which modern society relies on for mission-critical functionality. IC and system security and trust concerns include threats related to the malicious insertion of Trojan circuits designed to act as silicon time bombs to disable an IC, intellectual property (IP) and IC piracy, untrustworthy third-party IPs, exfiltrating sensitive material from an IC, and malicious system disruption and diversion.

Systems should be made trustworthy and secure hardware up. Security and trust should be considered as a first class metric during all stages in the IC and system design flows side-by-side power, cost, and performance. This special issue showcases advances in state-of-the-art trustworthy ICs and systems, and spans several aspects of IC and system security ranging from theoretical and conceptual foundations, synthesis, testing, and verification, through modeling and optimization, to case studies. From a trustworthy hardware platforms perspective this special issue covers IC and system security and trust issues in application-specific integrated circuits (ASICs), custom off the shelf (COTS), field-programmable gate arrays (FPGAs), microprocessors, and embedded systems. It explains novel hardware security primitives such as physical unclonable functions (PUFs), public PUFs (PPUFs), and random number generators. It presents design for trust techniques such as IC watermarking, fingerprinting, obfuscation, and split manufacturing. It presents

**This special issue showcases advances in state-of-the-art trustworthy integrated circuits and systems spanning a range of topics from theoretical and conceptual foundations, synthesis, and testing to modeling and optimization, and case studies.**

hardware-based security protocols for digital rights management (DRM), IC metering, enabling and disabling, and authentication. Finally, it presents a variety of hardware-based attacks.

The first two papers survey novel hardware-based security primitives. Identification, authentication, and integrity checking are important tasks for ensuring the security and protection of devices, programs, and data. The use of microscopic, random, and unclonable disorder in physical media for such security tasks has recently gained attention. The first paper “Physical unclonable functions and applications: A tutorial” is a tutorial on ongoing work in physical-disorder-based security, security analysis, and implementation choices. The second paper entitled “Public physical unclonable functions” surveys the time-bounded or public PUFs, including their design and security evaluation and the new protocols that they can support.

The following three papers focus on embedded systems security. There is a steady rise of interconnected embedded systems as part of the emerging vision of pervasive computing. The PC-centric Internet is evolving into the Internet of Things. Securing such networked embedded devices is vital. A prominent attack on networked embedded systems is the Stuxnet virus on the nuclear reactor controllers. Another example is the attacks on implanted and wearable

medical embedded devices. These devices introduce vulnerabilities that may, in the best case, compromise the privacy of a patient and in the worst case may be life threatening. Other examples include hacking critical automotive functions and street light controllers. The paper titled “Microcontrollers as (in)security devices for pervasive computing applications” discusses threats to microcontroller-based embedded systems, using two detailed case studies. The paper titled “Trustworthiness of medical devices and body area networks” surveys the threat landscape of medical embedded devices and the merits and shortcomings of existing defenses. The next paper in embedded security titled “Mobile trusted computing” surveys the trusted computing features in historical computing systems as well as in state-of-the-art mobile computing systems by focusing on features such as hardware support for platform integrity, attestation, secure storage, isolated execution, authentication, and provisioning.

The next two papers are on attacks on ICs. Counterfeiting of ICs has been on the rise, impacting the security and reliability of electronic systems. Reports show that recovered ICs are a significant percent of all counterfeit ICs in the market. Such ICs are recovered from scrapped boards of used devices. Identification of such counterfeit ICs is a challenge since these ICs are identical in appearance and functionality and package to fresh ICs. The paper titled “Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain” surveys the state of the art in counterfeiting and detection technologies. Another attack on ICs entails insertion, deletion, or malicious modifications in ICs. This attack can be launched at multiple points in the supply chain, ranging from the third-party IP cores to the foundry that fabricates the ICs. The paper “Hardware Trojan attacks: Threat analysis and countermeasures” provides a

comprehensive survey of the state-of-the-art Trojan attacks, modeling, and countermeasures.

Using FPGAs, a designer can separate the design process from the manufacturing flow. Sensitive designs need not be exposed to possible theft and tampering during their manufacture. However, there are other vulnerabilities introduced by FPGAs, such as the ability of an attacker to change the design by modifying or replacing the FPGA programming bit stream. The paper titled “FPGA security: Motivation, features, and applications” discusses all aspects of FPGA security and trust.

While hardware security and trust is a relatively recent concern, a somewhat similar yet fundamentally different problem of manufacturing defects has been extensively explored in the adjacent domain of very large scale integration (VLSI) test. Whereas the IC level attacks are man made, intentional, and hidden, manufacturing defects are unintentional. The paper titled “Regaining trust in VLSI Design: Design-for-trust techniques” surveys how concepts in VLSI test, such as fault analysis and delay tests, can be adopted in the context of hardware security and trust. Inspired by the design-for-testability (DfT) for better testability of manufacturing defects, this paper reviews design-for-trust (DfTr) solutions.

The final paper in this special issue, “A primer on hardware security: Models, methods, and metrics,” presents a novel primer on hardware security threat models, metrics, and remedies. The existing literature in this domain, however rich, assumes *ad hoc* models, solutions, and metrics, which lead to difficulty in analysis and comparison of the methodologies. This paper discusses the first of this kind classification of the underlying threat models, state-of-the-art defenses, and evaluation metrics for the following important hardware-based vulnerabilities: hardware malware (Trojan), reverse engineering,

side channels, counterfeiting, overbuilding, and/or piracy of the pertinent IPs. The classifications provide a guide for the academics and practitioners to clearly state the assumptions and problems, while fairly evaluating their solutions and building stronger countermeasures. The new systematization of knowledge that is provided by this work will pave the way for a more rapid and organized progress in this field.

In a nutshell, the increasing number of security threats and the cost of compromised systems are elevating security to be a first-order design requirement in computer systems side by side the requirements of speed, power, cost, and programmability. Consequently, similar to the design for manufacturability and design for testability mantras of the past, design for trust should be the new mantra to combat this very important threat. ■

## Acknowledgment

The guest editors would like to thank the contributors to this special issue and the reviewers—thought leaders and domain experts in hardware security—for making the guest editors’ task much easier and enjoyable notwithstanding the tight timeline. Special thanks go to Dr. C. Wang, Division Director in the U.S. Army Research Office (ARO), for helping organize a workshop on trustworthy hardware in New York City in fall 2013; several of the papers were derivatives of the presentations at this workshop. The guest editors would also like to thank the Managing Editor Vaishali Damle and the Publications Editor of the PROCEEDINGS OF THE IEEE Jo Sun for helping manage this special issue. The work of the guest editors was supported in part by NYU/NYU-AD Center for Research in Information Security Studies and Privacy (CRISSP); the National Science Foundation (NSF) under Grants 1319841, 1059328, and 0966187; and ARO.

## ABOUT THE GUEST EDITORS

**Ramesh Karri** received the Ph.D. degree in computer science from the University of California at San Diego, La Jolla, CA, USA.

He is a Professor of Electrical and Computer Engineering at the Polytechnic School of Engineering, New York University, Brooklyn, NY, USA. His research interests include trustworthy integrated circuits (ICs) and processors, high assurance nanoscale IC architectures and systems, very large scale integration (VLSI) design and test, and interaction between security and reliability. He has over 150 journal and conference publications in these areas. He is the Area Director for Cyber Security of the NY State Center for Advanced Telecommunications Technologies at NYU-Poly, Hardware Security Lead of the Center for Research in Interdisciplinary Studies in Security and Privacy (CRISSP, <http://crissp.poly.edu/>), cofounder of the Trust-Hub (<http://trust-hub.1102.org/>), and organizer of the annual red team blue team event at NYU, the Embedded Systems Security Challenge (<http://esc.isis.poly.edu/>).

Prof. Karri was the recipient of the Humboldt Fellowship and the National Science Foundation CAREER Award, and Best Student Paper Awards at the 2013 ACM Conference on Computer and Communications Security, the 2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, and the 2011 IEEE VLSI Design Conference. He cofounded and served as the Chair of the IEEE Computer Society Technical Committee on Nanoscale Architectures. He is a cofounder and steering committee member of the IEEE/ACM Symposium on Nanoscale Architectures (NANOARCH), Program Chair (2012) and



General Chair (2013) of the IEEE Symposium on Hardware Oriented Security and Trust (HOST), Program Co-Chair (2012) and General Co-Chair (2013) of the IEEE Symposium on Defect and Fault Tolerant VLSI and Nanotechnology Systems, and the General Chair of the 2013 NANOARCH. He serves on several program committees, including VTS, DAC, HOST, and ICCD. He is the Associate Editor of the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN, and the *ACM Journal of Emerging Technologies in Computing*. He is an IEEE Computer Society Distinguished Visitor and has organized invited tutorials on trustworthy hardware (including at the 2012 VLSI Test Symposium, the 2012 International Conference on Computer Design, the 2013 IEEE North Atlantic Test Workshop, the 2013 Design Automation and Test in Europe, the 2013 International Test Conference, and the 2014 IEEE/ACM Design Automation Conference).

**Farinaz Koushanfar** received the B.S. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1998, the M.S. degree from the University of California Los Angeles (UCLA), Los Angeles, CA, USA, and the M.A. degree in statistics and the Ph.D. degree in electrical engineering from the University of California Berkeley, Berkeley, CA, USA, in 2005.

She is an Associate Professor of Electrical and Computer Engineering (ECE) at Rice University, Houston, TX, USA. She is the Director of the Adaptive Computing and Embedded Systems (ACES) Laboratory.



# Physical Unclonable Functions and Applications: A Tutorial

*This paper is a tutorial on ongoing work in physical-disorder-based security, security analysis, and implementation choices.*

By CHARLES HERDER, MENG-DAY (MANDEL) YU, FARINAZ KOUSHANFAR, AND SRINIVAS DEVADAS, *Fellow IEEE*

**ABSTRACT** | This paper describes the use of physical unclonable functions (PUFs) in low-cost authentication and key generation applications. First, it motivates the use of PUFs versus conventional secure nonvolatile memories and defines the two primary PUF types: “strong PUFs” and “weak PUFs.” It describes strong PUF implementations and their use for low-cost authentication. After this description, the paper covers both attacks and protocols to address errors. Next, the paper covers weak PUF implementations and their use in key generation applications. It covers error-correction schemes such as pattern matching and index-based coding. Finally, this paper reviews several emerging concepts in PUF technologies such as public model PUFs and new PUF implementation technologies.

**KEYWORDS** | Arbiter; index-based coding; pattern matching; physical unclonable function (PUF); public model PUFs; ring oscillator; SRAM; unclonable

## I. INTRODUCTION

Mobile and embedded devices are becoming ubiquitous, interconnected platforms for everyday tasks. Many such tasks require the mobile device to securely authenticate

and be authenticated by another party and/or securely handle private information. Indeed, smartphones have become a unified platform capable of conducting financial transactions, storing a user’s secure information, acting as an authentication token for the user, and performing many other secure applications. The development of powerful mobile computing hardware has provided the software flexibility to enable convenient mobile data processing. However, comparable mobile hardware security has been slower to develop. Due to the inherent mobility of such devices, the threat model must include use cases where the device operates in an untrusted environment and the adversary has a degree of physical access to the system.

The current best practice for providing such a secure memory or authentication source in such a mobile system is to place a secret key in a nonvolatile electrically erasable programmable read-only memory (EEPROM) or battery-backed static random-access memory (SRAM) and use hardware cryptographic operations such as digital signatures or encryption. This approach is expensive both in terms of design area and power consumption. In addition, such nonvolatile memory is often vulnerable to invasive attack mechanisms. Protection against such attacks requires the use of active tamper detection/prevention circuitry which must be continually powered.

Physical unclonable functions (PUFs) are a promising innovative primitive that are used for authentication and secret key storage without the requirement of secure EEPROMs and other expensive hardware described above [7], [34]. This is possible, because instead of storing secrets in digital memory, PUFs derive a secret from the physical characteristics of the integrated circuit (IC). For example, this paper will discuss a PUF that uses the innate manufacturing variability of gate delay as a physical characteristic from which one can derive a secret. This approach is

Manuscript received September 3, 2013; accepted April 8, 2014. Date of publication May 30, 2014; date of current version July 18, 2014.

**C. Herder** and **S. Devadas** are with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Department of Electrical Engineering and Computer Science (EECS), Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: cherder@mit.edu).

**M.-D. Yu** is with Verayo, Inc., San Jose, CA 95129 USA and also with the Computer Security and Industrial Cryptography (COSIC) research group, KU Leuven, Leuven-Heverlee B-3001, Belgium.

**F. Koushanfar** is with the Adaptive Computing and Embedded Systems Lab (ACES), Department of Electrical and Computer Engineering (ECE), Rice University, Houston, TX 77005 USA.

Digital Object Identifier: 10.1109/JPROC.2014.2320516

0018-9219 © 2014 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See [http://www.ieee.org/publications\\_standards/publications/rights/index.html](http://www.ieee.org/publications_standards/publications/rights/index.html) for more information.

advantageous over standard secure digital storage for several reasons.

- PUF hardware uses simple digital circuits that are easy to fabricate and consume less power and area than EEPROM/RAM solutions with antitamper circuitry. In addition, simple PUF applications do not require expensive cryptographic hardware such as the secure hash algorithm (SHA) or a public/private key encryption algorithm.
- Since the “secret” is derived from physical characteristics of the IC, the chip must be powered on for the secret to reside in digital memory. Any physical attack attempting to extract digital information from the chip, therefore, must do so while the chip is powered on.
- Invasive attacks are more difficult to execute without modifying the physical characteristics from which the secret is derived. Therefore, continually powered active antitamper mechanisms are not required to secure the PUF [4].
- Nonvolatile memory is more expensive to manufacture. EEPROMs require additional mask layers, and battery-backed RAMs require an external always-on power source.

A PUF is based on the idea that even though the mask and manufacturing process is the same among different ICs, each IC is actually slightly different due to normal manufacturing variability. PUFs leverage this variability to derive “secret” information that is unique to the chip (a silicon “biometric”). In addition, due to the manufacturing variability that defines the secret, one cannot manufacture two identical chips, even with full knowledge of the chip’s design. PUF architectures exploit manufacturing variability in multiple ways. In addition to gate delay, architectures also use the power-on state of SRAM, threshold voltages, and many other physical characteristics to derive the secret.

This paper discusses the most popular PUF architectures. After defining a conceptual model for a PUF, this paper defines protocols to address two primary applications: strong authentication and cryptographic key generation. It then provides a case study on the most popular approach to each of these respective applications. It focuses on the experimental results from extensively tested realizations of actual PUF architectures that currently implement these protocols in real-world applications. Finally, this paper provides a perspective on future research relating to PUFs by briefly discussing current open problems as well as the latest proposed solutions.

### A. Previous Work: Unique Objects

Although many of the architectures that integrate PUFs into existing IC technology are new, it should be noted that the concepts of unclonability and uniqueness have been used extensively in the past for other applications [13]. For example, “unique objects” are well defined as objects with a unique set of properties (a “fingerprint”) based on the

unique disorder of the object [34]. This fingerprint should be stable over time and robust to other environmental conditions and to readout. Further, it must be “unclonable” in the sense that the cost to engineer and manufacture another object with the same fingerprint must be prohibitively expensive or impractical using known manufacturing techniques (including by the original manufacturer).

One example of early usage of unique objects for security was proposed for the identification of nuclear weapons during the Cold War [9]. One would spray a thin coating of randomly distributed light-reflecting particles onto the surface of the nuclear weapon. Since these particles are randomly distributed, the resulting interference pattern after being illuminated from various angles is unique and difficult to reproduce.

Therefore, immediately after being applied, each interference pattern would be measured as a “signature” and stored in a secure database. A weapon could then be identified at any later time by re-illuminating the surface and comparing the interference pattern against the measured interference pattern. At the time, it was presumed to be infeasible to reproduce such an interference pattern even if an adversary knew the illumination angle(s) and the resulting pattern.

## II. TYPES OF PUFs

The two primary applications of PUFs are for: 1) low-cost authentication; and 2) secure key generation. These two applications have resulted from the fact that PUFs designed during the past decade have mostly fallen into two broad categories. These categories are described as “strong PUFs” and “weak PUFs.” Strong PUFs are typically used for authentication, while weak PUFs are used for key storage.

Each PUF can be modeled as a black-box challenge–response system. In other words, a PUF is passed an input challenge  $c$ , and returns a response  $r = f(c)$ , where  $f(\cdot)$  describes the input/output relations of the PUF. The black-box model is appropriate here, because the internal parameters of  $f(\cdot)$  are hidden from the user since they represent the internal manufacturing variability that the PUF uses to generate a unique challenge–response set. Such parameters would include the variability of a circuit’s internal gate delay as described in the Introduction. PUF security relies on the difficulty of measurement or estimation of these parameters as well as the difficulty of manufacturing two chips with the same set of parameters.

The fundamental difference between weak and strong PUFs is the domain of  $f(\cdot)$ , or informally, the number of unique challenges  $c$  that the PUF can process. A weak PUF can only support a small number of challenges (in some cases only a single challenge). A strong PUF can support a large enough number of challenges such that complete determination/measurement of all challenge–response pairs (CRPs) within a limited timeframe is not feasible.

### A. Weak PUF Model

The first class of PUFs leveraging manufacturing variability are weak PUFs [also known as physically obfuscated keys (POKs)]. These PUFs can be thought of as PUFs that directly digitize some “fingerprint” of the circuit. This direct measurement results in a digital signature that can be used for cryptographic purposes. Because the fingerprint signature remains largely invariant, this means that the PUF can only be interrogated by one or a small number of challenges. In the above black-box description, this corresponds to  $f(\cdot)$  having a domain of one or only a small number of inputs. Correspondingly,  $f(\cdot)$  will also have a very small range, as a given challenge should always result in the same response (ignoring noise, which is considered later). One can clearly use several instances of the above black box to support more CRPs or response bits. However, this is still considered a weak PUF, because the number of responses is linearly related to the number of components subject to manufacturing variation. Explicitly stated, weak PUFs have the following properties:

- a small number of CRPs (linearly related to the number of components whose behavior depends on manufacturing variation);
- response is stable and robust to environmental conditions and multiple readings so that a challenge always yields the same response;
- responses are unpredictable and depend strongly on the innate manufacturing variability of the device;
- it is impractical to manufacture two devices with the same physical fingerprint.

An example weak PUF is the power-on state of an SRAM. Although a SRAM cell is symmetric, manufacturing variability will give each cell a tendency toward a logical “1” or “0” at power-on. This variability is random across the entire SRAM, giving it a unique fingerprint on power-on that can be identified. In this case, if the “response” consists of the entire SRAM state at power-on, the notion of a “challenge” is not useful, as there is only one possible “challenge”: powering on the SRAM. The output signature is always the same (ignoring noise). One can allow for more output bits by increasing the size of the SRAM, but the response space is still linearly related to the number of components subject to manufacturing variation (each SRAM cell). The SRAM is an extreme example of a weak PUF in the sense that it only has one “CRP.”

Note that since weak PUFs in general have only a small number of CRPs, these pairs must be kept secret. If a weak PUF only has one CRP, and it is revealed, then any device can emulate the PUF. For this reason, weak PUFs are well suited for use in key derivation processes. The PUF provides the randomness and secure storage, and the secret key (derived from the PUF’s response bits) is never revealed during operation.

Once the key is recovered by the PUF (this typically requires error correction), any cryptographic process may follow. For example, the weak PUF output may be used as

the key in a keyed-hash message authentication code (HMAC) challenge–response sequence. In addition, the output may be used as a secret key to encrypt/decrypt data on the device.

### B. Strong PUF Model

Strong PUFs differ from weak PUFs in that a strong PUF can support a large number of CRPs. As a result, a strong PUF can be authenticated directly without using any cryptographic hardware. The requirements for a strong PUF are:

- large enough challenge–response space such that an adversary cannot enumerate all CRPs within a certain fixed time (ideally, exponential in the number of challenge bits);
- responses stable to environment, multiple readings;
- an adversary given a polynomial-sized sample of adaptively chosen CRPs cannot predict the response to a new, randomly chosen challenge;
- not feasible to manufacture two PUFs with the same responses;
- the readout only reveals the response  $r = f(c)$  and no other data about the internal functionality of the PUF.

It should be noted that a weak PUF can provide authentication capabilities if the weak PUF is paired with crypto hardware supporting HMAC or similar authentication processes (note that HMAC and others support exponentially sized challenge–response spaces but their use requires 100% response stability and, therefore, error-correction logic). It should also be noted that the security models for weak and strong PUFs differ. The output of a weak PUF must be kept private, while a strong PUF’s responses do not have the same restriction.

The strong PUF has the additional requirement of readout access restriction [only  $r = f(c)$  is revealed] due to this difference in security models. In addition, to prevent total enumeration of the strong PUF, one must also consider the readout time of the PUF in conjunction with the number of CRPs. A faster PUF response allows for faster enumeration of all PUF CRPs. Since a weak PUF provides a secret key, the surrounding digital cryptographic hardware is responsible for limiting access to the weak PUF output. However, the strong PUF does not require the use of additional crypto hardware to provide authentication services, and therefore must itself prevent unauthorized access into its own internal structure.

### C. Error Correction

Both weak and strong PUFs rely on analog physical properties of the fabricated circuit to derive secret information. Naturally, these analog properties have noise and variability associated with them. Consider the example introduced in Section I that uses gate delay. This delay depends on temperature, supply voltage, and other environmental parameters. As these parameters vary, so does

the “digital fingerprint” measured by the PUF. If the parameters vary too much, the digital key (for the weak PUF) or response (for the strong PUF) will change, and the crypto operation will fail.

The first mechanism to mitigate such effects is to use differential design techniques to cancel out first-order environmental dependencies. Using the gate delay example, typical PUFs using this effect will not measure a single gate’s delay, but rather the difference between two identically designed, but distinct gates on a die. In this way, any environmental factor should affect each gate equally.

Although differential design methodologies do improve reliability, noise is still a factor in PUF design. Even in optimal environmental conditions, noise will result in one or several of the output bits of the PUF being incorrect for any given challenge. Therefore, modern PUF designs employ multiple error-correction techniques to correct these bits, improving reliability. However, many of these error-correcting techniques have been shown to leak bits of the secret key, since they require the computation and public storage of syndrome bits. As such, an excess number of PUF bits are generated and then downmixed to produce a full entropy key.

In addition to standard error-correction techniques, PUFs also use soft-decision coding. This coding technique takes advantage of the reliability information of a given response bit to improve error-correction performance. This reliability information can be obtained from repeated PUF response readings in the case of SRAM PUFs, or the magnitude of frequency difference values in the case of ring-oscillator PUFs. Both of these error-correcting techniques will be discussed further in the context of the weak and strong PUF examples to be presented in Sections IV-E, VI-B, VI-C1, and VI-D1.

### III. EXAMPLE STRONG PUF ARCHITECTURES

#### A. Optical PUF

One of the first implementations of a strong PUF was constructed by Pappu *et al.* in 2001 [30]. The paper terms the device a “physical one-way function,” but the functionality is identical to that of a strong PUF. Pappu *et al.* describe a device with three primary components: 1) a laser directed along the Z-axis that can be moved in the XY-plane and whose polarization can be modified; 2) a stationary scattering medium that sits along the path of the laser beam; and 3) an imaging device that records the output “speckle” pattern of laser light exiting the scattering medium.

In this device, the input challenge is a laser XY location and polarization, and the response is the associated speckle pattern. The speckle pattern is strongly dependent on the input location/polarization because multiple scattering events occur inside the scattering medium. In the imple-

mentation by Pappu *et al.*, the scattering medium consisted of a large number of randomly positioned 100- $\mu\text{m}$  silica spheres suspended in a hardened epoxy. Each sphere acts as a small lens, refracting individual rays of light as they move through the scattering block. The overall size of the scattering block was on the order of 1-mm thickness. Therefore, even a relatively simple optical path must encounter  $\sim 10$  spheres as it travels through the scattering block.

All of these paths then are focused into an image on the detector. It is intuitively true that each of these paths will be very sensitive to input coordinates. Studies on speckle patterns produced by reflection/transmission by rough surfaces have found this to be true both experimentally and mathematically [2]. In addition, the speckle pattern is also sensitive to the internal structure of the scattering block. Therefore, it is difficult to fabricate two blocks with identical speckle patterns. Finally, due to the complex nature of the physical interactions, it is difficult to model the internal dynamics of the scattering medium. It is also difficult to use the output speckle to determine properties of the scattering block (such as the locations of the silica spheres).

These assumptions, while not strictly based on known computationally difficult problems, can be trusted to be difficult due to the fact that ray-tracing electromagnetic simulation is a well-studied field with established theoretical models and best practices. One can make the statement that if an adversary were able to break the above optical PUF by efficiently reproducing the physical device, modeling the entire scattering block, or discovering the sphere locations via observation of the speckle pattern, this would represent a major advancement in the field of ray-based models of electromagnetic simulation. It is for this reason that Pappu *et al.* described this optical PUF as a “physical one-way function.”

#### B. Arbiter PUF

Although the capabilities of the above optical PUF are significant, and they represented a significant step forward in the understanding and construction of PUFs, the practical applications are limited due to the macroscopic optical nature. This limitation stemmed from two properties.

First, the actual unclonable object (the scattering block) was separate from the measurement apparatus (the imaging device). As a result, the trust gained from authenticating an optical PUF is more limited. In a practical use case, the objective of authenticating the PUF is typically to authenticate the associated processor to which it is connected. However, since the optical PUF is separated from the digital measurement circuitry, an optical PUF as described by Pappu *et al.* designed to authenticate processor A can easily be detached from processor A and connected to processor B. Processor B could then authenticate itself as processor A. It is more desirable for the digital measurement apparatus to be integrated in with the

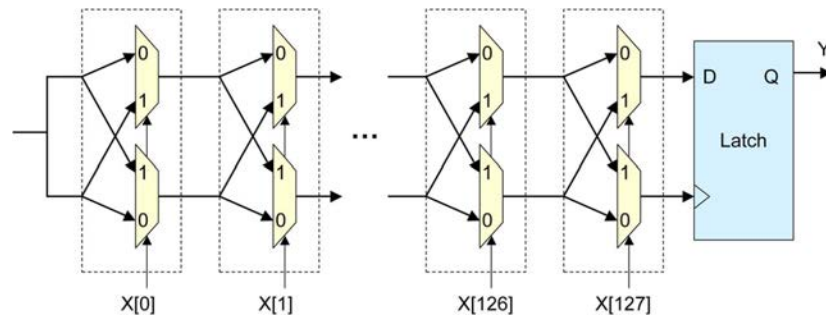


**Fig. 1. Arbiter PUF circuit.** The circuit creates two delay paths with the same layout length for each input  $X$ , and produces an output  $Y$  based on which path is faster.

PUF such that the PUF is not separable from the device it is used to authenticate.

Second, since both key generation and authentication applications use integrated electronics, a more practical PUF would have the same properties as the optical PUF and simultaneously be integrated directly with a conventional complementary metal–oxide–semiconductor (CMOS) process. This integration would be such that the IC could not be separated from the PUF.

Silicon implementations of strong PUFs were described in the paper by Gassend *et al.* beginning in 2002 using manufacturing variability in gate delay as the source of unclonable randomness [7]. In one implementation, a race condition is established in a symmetric circuit. This is shown in Fig. 1. An input edge is split to two multiplexors (muxes). Depending on the input challenge bits ( $X[0]–X[127]$ ), this path will vary. Although the layout is identical (propagation time should be the same for each edge no matter what challenge bits are chosen), manufacturing variability in the gate delay of each mux will result in one edge arriving at the latch first, and the latch acts as the “arbiter.” The output will, therefore, depend on the challenge bits.

In Fig. 1, there are 128 challenge bits and one response bit. Of course, one typically operates multiple identical circuits in parallel to achieve 128 response bits. In this way, the arbiter PUF can be scaled to an almost arbitrary number of CRPs.

The security of the arbiter PUF, like the optical PUF before it, is based on assumptions regarding manufacturing capabilities and ultimately metrology of the individual gate delays. Because the design is symmetric, the design does not contain any “secret” information. An adversarial manufacturer that has the PUF design cannot manufacture a duplicate PUF, because the behavior of the PUF is defined by the inherent variability in the manufacturing process. Even the original manufacturer of the PUF could not produce two identical PUFs, since this would require a significant improvement in manufacturing control.

The second security assumption is that the individual gate delays are difficult to measure directly. It assumes that

an invasive attacker would have difficulty in extracting the individual delays even with physical access. This assumption is based on the hypothesis that an invasive attacker would destroy the gate delay properties using his/her measurement techniques.

The last security assumption is that given a set of CRPs from an arbiter PUF, an adversary could not calculate the internal delays of the gates. For the architecture described above, this is actually not the case. Each delay is independent from all other delays, and the delays add linearly. As a result, one can use standard linear system analysis to intelligently gather data about the gate delays from the response bits. In fact, it can be shown that this system breaks after only a small number of challenges [17]. This problem can be resolved by several approaches proposed by Gassend *et al.* and described in Section IV-D.

Finally, in both optical and arbiter PUF architectures, it should be noted that environmental factors play a significant role. For the optical PUF, calibration of the input location is a concern. In the case of the arbiter PUF, one can easily recognize that environmental variations such as temperature, supply voltage, aging, and even random noise will affect the delay of each edge through the arbiter PUF. In addition, if the delays are close enough, the latch’s setup time will be violated, potentially resulting in an unpredictable output. As a result, the response bits may not be stable. In this case, error-correcting techniques are used to increase the stability of the PUF while maintaining its security. Techniques for accomplishing this will be covered in Section IV-E. Although key generation has zero error tolerance, PUF authentication usually incorporates an allowable error threshold, thereby decreasing the stability requirement, and often obviating the need for error correction.

#### IV. LOW-COST AUTHENTICATION: STRONG PUFs

The strong PUF architectures described above are typically associated with the application of low-cost authentication.

In this case, a strong PUF will replace the secure memory and crypto hardware on an embedded device and is used to securely identify the device to a server. Because the PUF does not require secure nonvolatile memory, antitamper circuitry, or additional supporting crypto acceleration hardware, a PUF-based solution requires less area, power, and mask layers than a traditional approach to secure authentication.

### A. Authentication Protocol

As described previously, the strong PUF receives a challenge and generates a response. However, the requirements of a strong PUF state that an adversary provided with polynomial CRPs should not be able to predict the response to a new challenge.

Although this is a desirable property, it also presents a usage problem. Since the PUF acts as a “black box,” even the authentication server only has access to previously observed CRPs and, therefore, also cannot predict the response to a new challenge.

Therefore, the protocol for using PUFs is significantly different than most public/private key cryptographic systems. Consider a server authenticating a client.

- 1) PUF is manufactured.
- 2) Server obtains access to PUF and generates a table of CRPs. These pairs are stored in internal secret storage.
- 3) PUF is given to the client.
- 4) The client submits a request to the server to authenticate.
- 5) Server picks a known CRP and submits the challenge to the client.
- 6) The client runs the challenge on the PUF, returns the response to the server.
- 7) Server checks to see that the response is correct and marks the CRP as used.

Because the server cannot predict the PUF behavior, it must internally store CRPs to be used later. Each CRP must be used only once. Therefore, the server must either store enough CRPs so that it will not run out, or it must periodically “recharge” the table by establishing secure communication with an authenticated client and requesting responses to new challenges. To address the CRP table scalability problem, newer protocols based on storage of a compact model for PUF have emerged. A brief discussion of these protocols is included in Section VII-A.

Note that each client PUF will have unique CRPs, and therefore can be individually authenticated. In addition, the server must store tables of CRPs for each of the clients to be authenticated.

### B. Arbiter PUF Topologies

The initial implementation of silicon PUFs had known security issues due to the fact that the delays were linearly added to produce the resultant response bit [8]. As a result, they could be learned with relative ease. This issue

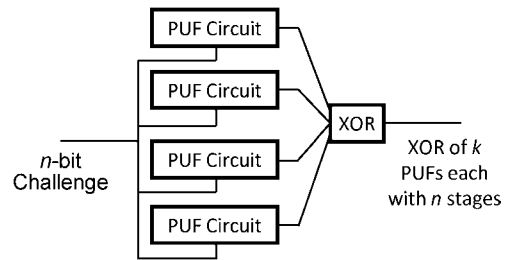


**Fig. 2. Four individual arbiter PUF circuits with nonlinearities introduced via xor'ing their outputs.**

naturally led to the introduction of other “nonlinear” effects to make such modeling attacks more difficult. These efforts included xor arbiter PUFs, lightweight secure PUFs, and feedforward arbiter PUFs [8], [16], [17], [22], [39].

In a xor arbiter, multiple arbiter PUF outputs are xor'ed to form a single response bit. This is shown in Fig. 2. These structures have shown greater resilience against machine learning attacks [24], [35]. Recent studies have demonstrated the vulnerability of the xor arbiters to a combination of machine learning and side-channel attacks [19], [36]. Developing methods to suppress the side channels could help in alleviating this vulnerability.

### C. Arbiter PUF Implementation

The arbiter PUF was implemented and studied by Devadas *et al.* as a part of a radio-frequency identification (RFID) IC fabricated in 0.18- $\mu\text{m}$  technology [4]. In this implementation, a single arbiter PUF is implemented on-chip. This primitive has an input challenge of 64 bits and a single output bit. To construct a  $k$ -bit response, a linear feedback shift register (LFSR) is used to generate a pseudorandom sequence based on the input challenge. The PUF is then evaluated  $k$  times using  $k$  different bit vectors from this larger pseudorandom sequence. Finally, to prevent learning attacks on the PUF output bits, an additional scrambling routine is performed.

In this implementation, area and power consumption represented a major design constraint. Therefore, the above PUF implementation with only a single arbiter is used. As a result, the majority of the silicon area is consumed by standard RFID components (RFID front-end, one-time programmable memory, digital logic). The PUF and associated LFSR have been implemented in less than 0.02  $\text{mm}^2$  using 0.18- $\mu\text{m}$  fabrication technology. In addition, the PUF only consumes dynamic power during evaluation, and the power consumption was shown to be small with respect to the power stored on the RFID chip.

In order to understand the PUF's utility as an identification and authentication source, intra-PUF and inter-PUF variation are defined as follows [7].

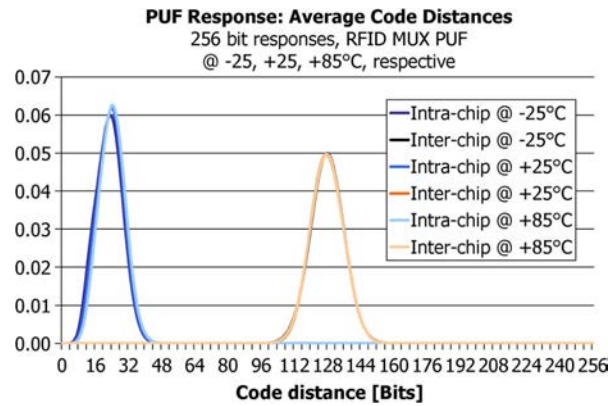


**Fig. 3.** Code distance distribution for 256-bit PUF responses.

- Intra-PUF variation: Defined as the number of bits in a PUF response that vary when an identical challenge is repeatedly queried on a given PUF device in a changing environment. This variation is due to this environmental change as well as statistical noise. As a result, it is commonly represented in the form of a statistical distribution. Intra-PUF variation is a measure of the reproducibility of responses from an individual PUF circuit.
- Inter-PUF variation: Defined as the number of bits in a PUF response that vary between different devices for a set of shared challenges. This is due to differences between the physical ICs and is also commonly represented in the form of a statistical distribution. The inter-PUF variation is a measure of the uniqueness of an individual PUF circuit.

For the application of secure authentication, intra-PUF variation should be low (ideally 0%) so that the PUF can be verified. On the other hand, inter-PUF variation should be high (ideally 50% on average) so that two separate PUFs have a maximally decorrelated responses. This behavior has been observed for arbiter PUFs, as shown in Fig. 3. Note that in this figure, the two interchip variations are roughly 50% (128 bits out of 256), and the intrachip variations are much smaller (~10%). Clearly, the implemented PUF has the desired properties. (However, it is also clear that error correction must be used to compensate for the ~10% intrachip variation for key generation applications.)

#### D. Attacks on Arbiter PUFs

The security of a strong PUF depends on several factors (if any one of these factors is compromised, the security of the PUF itself is also compromised):

- difficulty of measurement of PUF internal parameters (only CRPs can be measured);
- difficulty of manufacturing “clones”;
- difficulty of predicting PUF behavior based on past CRPs.

For arbiter PUFs, an area of continued active research is in ensuring the difficulty of predicting PUF behavior based on past CRPs. In the RFID IC example, this issue is addressed by “scrambling” the output bits of the PUF. In other words, the output bits of the PUF pass through some digital circuit that obfuscates the linear behavior of the PUF before being returned as a response. The simple arbiter PUF implementation without output postprocessing is linear and, therefore, significantly easier to predict. (We note that this scrambling will increase the noise in the output bits and, therefore, has to be done carefully.)

Studies have been performed by Majzooobi *et al.* [23] and Rührmair *et al.* [35] using machine learning to predict the behavior of PUFs after a certain number of CRPs have been observed. In these studies, learning attacks were perpetrated on simple arbiter PUFs, feedforward arbiter PUFs, arbiter PUFs with output xor’ing, “lightweight secure” arbiter PUFs (these use a more complicated output postprocessing circuit, but are based on principles similar to output xor’ing).

The linear behavior of simple arbiter PUFs was clearly demonstrated, as a learning algorithm predicted the behavior of a 64-bit arbiter PUF with 95% accuracy after observing 640 CRPs (the model training time on a standard PC was 0.01 s). To predict with 99.9% accuracy, 18 050 CRPs are needed to be observed (model training time of 0.6 s). This demonstrates that the behavior of a simple arbiter PUF can be learned efficiently [35].

In addition, PUFs with 64-bit challenges and 128-bit challenges were tested. It was found that the number of CRPs and model learning complexity scaled as expected with the input challenge size. This proved to be true not just for simple arbiter PUFs, but also for the nonlinear implementations discussed below.

The arbiter PUF with output xor’ing, on the other hand, is able to make the problem intractable to such learning attacks. Rührmair *et al.* identified an exponential dependence on the number of output xors and the required complexity of the learning attack. This is due in part to the actual computational complexity of learning the model. It is also due to the fact that this model learning process is looking for a global optimum on a nonconvex parameter space. As a result, the learning algorithm has some probability of failure that scales inversely with the size of the training set (the number of observed CRPs). If the algorithm fails, it must be restarted again with different parameters.

It was identified that an arbiter PUF with a 512-bit challenge and eight output xors would defeat the machine learning approaches used by Rührmair *et al.* in 2010. The postprocessing scheme used in “lightweight secure” PUFs proved to have the same exponential dependence with a similar complexity requirement.

A newer set of attacks leveraging both machine learning and side-channel information has recently emerged [19], [36]. It has been shown that by coordinated

application of timing or power side-channel analysis and adapted machine learning techniques, very efficient attacks can be performed, i.e., attacks that use linearly many CRPs and low degree polynomial computation times. The practical viability of the combined attacks has been demonstrated by machine learning experiments on numerically simulated CRPs. This work has shown that XOR arbiter PUF and lightweight PUFs have to be implemented in such a manner that power side channels are protected, else PUFs can be easily cloned.

One key consideration in studying the complexity of PUFs is stability (described as “intra-PUF variation” in the RFID IC example). This will be discussed further in Section IV-E. However, it should be noted that although output XOR’ing has an exponential effect on modeling complexity, it also has an exponential effect on decreasing stability. In doing so, it decreases the effectiveness of a PUF in an actual authentication environment and simultaneously decreases the accuracy requirement of an attack model, as the greater intrinsic PUF error must be tolerated by the authentication protocol.

The task of identifying an approach to exponentially increase model complexity while only having a polynomial effect decreasing PUF stability is an area of active research.

### E. Error Correction Versus Tolerance

In the RFID IC example, random noise contributes to the PUF stability being roughly 90%, i.e., the intra-PUF variation is  $\sim 10\%$ . In addition, this stability worsens when the temperature changes.

In perhaps the earliest reference to error correction in silicon PUFs, Gassend mentioned the use of 2-D Hamming codes [6]. Suh suggested the use of Bose–Chaudhury–Hochquenghen (BCH) codes—more specifically the BCH (255,63,  $t = 30$ ) code [38]. In this case, the PUF generates 255 bits, but the code exposes 192 syndrome bits publicly, so the actual security of the system is at most 63 bits. This error corrects at most 30 errors out of 255 bits. This corresponds to a PUF with  $\sim 88\%$  stability.

In low-cost authentication applications, the host instead gives a certain error tolerance or multiple authentication opportunities to a PUF before rejecting the PUF as invalid. Error tolerance is typically the preferred methodology. Using the arbiter PUF described with code distances shown in Fig. 3, false positive/negative identification probabilities were measured for specific allowed error tolerances. These data are summarized in Fig. 4.

Majzoobi *et al.* showed a strong PUF remote authentication protocol which does not require traditional error correction while also being secure against machine learning attacks [25], [31]. The protocol is inspired by the pattern matching techniques described in Section VI-B but is used for authentication rather than secret key generation. A large class of machine learning attacks are shown to fail when applied to this protocol.

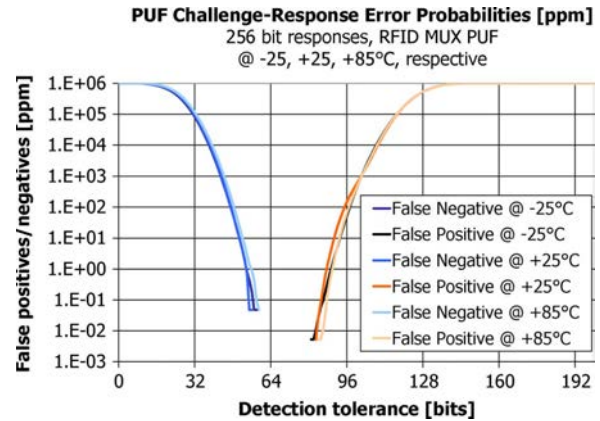


Fig. 4. False positives and negatives for strong PUF operation with a given error tolerance.

## V. EXAMPLE WEAK PUF ARCHITECTURES

### A. Ring-Oscillator PUF

In addition to arbiter PUFs, the manufacturing variability intrinsic to circuit gate delay can also be used to instantiate a “ring-oscillator PUF” [39]. This PUF architecture contains  $N$  identically designed ring oscillators synthesized onto a field-programmable gate array (FPGA) or an application-specific integrated circuit (ASIC).

Due to the variation in delay of the inverters in the ring oscillator, each will have a slightly different frequency. The frequencies of two oscillators are measured and compared to reveal one of the PUF output bits. If there are  $N$  oscillators, there are  $N(N - 1)/2$  possible pairings. However, the number of output bits is limited due to correlations (if ring oscillator A is faster than B, and B is faster than C, then clearly A is faster than C). For  $N$  oscillators, there is a specific ordering of fastest to slowest. If the oscillators are truly identical and manufacturing variation dominates, then each of these  $N!$  orderings is equally likely. Therefore, there are a maximum of  $\log(N!)$  bits that can be extracted from the PUF.

Note that the ring-oscillator PUF is a weak PUF, since there are a limited number of “challenge bits” that can configure the PUF’s operation. Once fabricated, the ring oscillators’ frequency is set, so the output bits of the PUF will always remain constant.

Because the ring-oscillator PUF measures differences in gate delay like the arbiter PUF, the ring-oscillator PUF is susceptible to the same set of environmental variations and noise sources. Therefore, error correction will be equally important in this application.

One approach that can be taken immediately to mitigate potential errors is to recognize that oscillators that are “close” in frequency have much greater likelihood of causing an output error than oscillators that are “far apart”

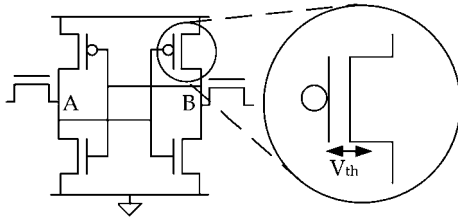


**Fig. 5. SRAM cell [12].**  $V_{th}$  differences in the transistors result in the SRAM powering up in either a logic “0” ( $A = 0, B = 1$ ) or logic “1” ( $A = 1, B = 0$ ).

in frequency. This is because the small fluctuations in oscillator frequency due to noise or environmental variations are less likely to cause a bit flip when the two oscillator frequencies are far apart [39].

Therefore, at the time of provisioning, one can select only pairs of oscillators whose frequencies are sufficiently far apart to define the PUF output bits. This increases the PUFs robustness toward noise and environmental variations.

## B. SRAM PUF

Both the arbiter PUF and the ring-oscillator PUF ultimately depend on variations in the propagation delay of gates. However, this is not the only physical property on which a PUF can be built. A popular weak PUF structure exploits the positive feedback loop in a SRAM or SRAM-like structure shown in Fig. 5. A SRAM cell has two stable states (used to store a 1 or a 0), and positive feedback to force the cell into one of these two states and, once it is there, prevent the cell from transitioning out of this state accidentally.

A write operation forces the SRAM cell to transition toward one of the two states. However, if the device powers up and no write operation has occurred, the SRAM cell exists in a metastable state where theoretically, the feedback pushing the cell toward the “1” state equals the feedback pushing the cell toward the “0” state, thereby keeping the cell in this metastable state indefinitely. In actual implementations, however, one feedback loop is always slightly stronger than the other due to small transistor threshold mismatches resulting from process variation. Natural thermal and shot noise trigger the positive feedback loop, and the cell relaxes into either the “1” or “0” state depending on this process variation.

Note that since the final state depends on the difference between two feedback loops, the measurement is differential. Therefore, common mode noise such as die temperature, power supply fluctuations, and common mode process variations should not strongly impact the transition.

Although this idea was patented in 2002, the first experimental implementation was performed in 2007, where a custom SRAM array based on 0.13- $\mu\text{m}$  technology was

shown to generate random values based on threshold mismatches [15], [37]. This work demonstrated a roughly normal distribution of bits and more than 90% bit stability. Additional work showed that SRAM initialization can produce a unique physical fingerprint for each chip [11], [12].

Like other strong and weak PUF implementations, the SRAM PUF is also sensitive to noise. If the two feedback loops of the SRAM cell are sufficiently close, then random noise or other small environmental fluctuations can result in an output bit flip. Therefore, error correction of this output will be necessary.

Like the ring-oscillator PUF, the architecture of the SRAM PUF can be used to make intelligent decisions regarding error coding. The key recognition is that the relative strengths of the two feedback loops in a SRAM cell are relatively static. A cell strongly biased toward “1” or “0” will remain strongly biased toward “1” or “0,” respectively. Therefore, by using repeated measurements, one can assess the stability of a SRAM PUF output bit and selectively use the most stable bits as the PUF output. This process is used in conjunction with traditional coding techniques to mitigate the noise inherent to SRAM PUFs.

## VI. CRYPTOGRAPHIC KEY GENERATION: WEAK PUFs

Due to their limited challenge–response space, weak PUF architectures are typically used for cryptographic key generation. In this case, a weak PUF will replace a secure nonvolatile memory that would have stored the cryptographic key. Once the key is derived from the weak PUF, it is stored in secure volatile memory during the device’s operation. This key can then be used for authentication, encryption, and other cryptographic protocols. Due to the fact that one or very few keys can be generated by the PUF, the security of this key during operation is of paramount importance. If the secure key is revealed, any device can emulate the weak PUF.

### A. Key Generation Protocol

Because weak PUFs like the ones discussed above have effectively fixed “challenge bits,” the key generation protocol is fairly simple. In the case of the SRAM PUF, one simply powers on the SRAM and observes the memory state. Similarly for the ring-oscillator PUF, one simply pairwise compares each of the oscillators in order to measure the correct ordering of oscillation frequency.

In both of these cases, the complexity lies in the limitations of physical implementations that result in both statistical and systematic noise that must be corrected/mitigated. The actual approach used to address these issues differs for SRAM and ring-oscillator PUFs because the underlying physical implementation is different.

Ultimately, a stable set of unique bits is extracted from the weak PUF. These bits can then be used in any of a number of cryptographic protocols. Note that weak PUFs

can be used for authentication (similar to strong PUFs) even though they do not have a large number of CRPs. By supplementing the weak PUF with a hardware HMAC/AES implementation, one can achieve authentication capability at the cost of the additional power and area required by the cryptographic hardware primitives that embody the HMAC/AES protocol.

## B. Arbiter PUF With Pattern Matching Error Correction

Although weak PUFs are typically used for secure key generation due to their limited challenge–response space size, protocols allowing for strong PUFs to be used in this capacity have also been developed. A key challenge in adapting strong PUFs to key generation is in correcting errors in PUF response bits. To this end, Paral and Devadas have proposed the use of a “pattern matching” technique to correct for errors in strong PUFs for use as a key generation mechanism [29].

A full description of this work is beyond the scope of this paper. In a nutshell, this approach reverses the traditional challenge–response format of a PUF. In this case, a secret offset  $I$  is chosen (the key is derived from  $I$ ). A  $W$  bit portion of the response at offset  $I$  is published publicly. To recover the key, a strong PUF iterates through a deterministic set of challenges (which may depend on previous secret data that have been measured). The PUF response to this challenge contains the  $W$  pattern bits in the output. The PUF uses an error-tolerant comparison circuit (up to  $T$  bits of error) to identify the offset  $I$  of the  $W$ -bit block in the PUF response bits. The secret key is then derived from  $I$ . This process can be repeated several times to obtain larger sets of secret bits.

The study identified that, with the correctly chosen parameters (PUF output size: 1024,  $W$ : 256,  $T$ : 80), the error occurrence could be decreased such that the PUF always succeeded in regenerating the key on the first try across environmental conditions.

## C. SRAM PUF Implementation

As previously mentioned, the SRAM PUF leverages the threshold voltage mismatch of transistors in a SRAM cell due to manufacturing mismatch. This mismatch results in a repeatable tendency to settle into a “1” or “0” state when the SRAM cell is powered on with no writes occurring. Several studies have constructed SRAM PUFs and analyzed their properties.

One of the first implementations of such a chip identification system was tested by Su *et al.* with RFID applications [37]. In this study, a custom SRAM cell was constructed to minimize potential systematic mismatch between the two transistors. Such a skew would result in a given SRAM cell being more likely to favor a “1” than a “0” or *vice versa*, even with random process variation. To prevent such systematic skew, they used analog layout tech-

niques to construct a “symmetric” and “common centroid” layout of the SRAM cell.

The study demonstrated that the SRAM PUF behaved as desired. After fabrication, an equal number of SRAM cells tended toward “1” and “0” to within experimental error for both layouts. The study identified that cell positioning within the SRAM, SRAM positioning on the wafer, and subsequent wafers were all decorrelated with the SRAM cell’s tendency toward “1” or “0.”

A challenge arose with the recognition that roughly 4% of the SRAM cells did not have enough mismatch to strongly favor “1” or “0.” These cells probabilistically settled into “1” or “0” at random due to the contributions of thermal and shot noise. The number of these unstable bits increased at temperature/voltage corners and as the chip aged.

The study by Holcomb *et al.* tested the functionality of SRAM PUFs on off-the-shelf RAM and processor products such as the MSP430 and Intel’s WISP RFID device [11]. In this application, the SRAM cell was a part of another on-chip SRAM that was actively used for program/data and not custom fabricated in any way to enhance stability or skew performance.

In this way, an end user can use an existing off-the-shelf component with no silicon modification and, using software alone, implement a weak PUF for cryptographic or identification purposes.

Although off-the-shelf SRAM cells are not optimized for usage as a PUF, Holcomb *et al.* did observe a bit stability of 5% across temperatures from 0 °C to 50 °C. This stability is roughly the same as the stability measured by Su *et al.*, indicating that the custom fabrication did not help significantly in this regard.

However, the off-the-shelf SRAM cells were observed to have a significant bias toward the “1” state. This changes the entropy and unique identification analysis. In this study, 512 B of SRAM were used as a fingerprint. Due to the systematic skew of the SRAM cells, the min-entropy of this block was roughly 200 bits (plus/minus 10 bits depending on temperature). This 512-B block was then passed through a universal hash to extract 128 bits of output data.

Finally, because the SRAM is being used as a memory element for the processor, it is always powered, even if the PUF section of the SRAM is never written during normal operation. This continual powering of the SRAM in a “1” or “0” state results in negative bias temperature instability (NBTI). This is a type of “burn-in” for deep submicrometer metal–oxide–semiconductor field-effect transistor (MOSFET) technology, where the threshold voltage of a transistor increases over time due to the applied stress conditions of high temperature and a constant vertical electric field across the gate terminal while the transistor is “on.”

Therefore, if a SRAM cell is powered on and set to the “0” state for a long time ( $\sim 10$  days), then on subsequent

power-on sequences, the cell is more likely to skew toward the “1” state. This predictable behavior stands in contrast to the behavior of temperature variations, which can either skew the cell toward “0” or “1” as the temperature fluctuates.

1) *SRAM PUF Error Correction*: Because these papers targeted the application of die identification, rather than key generation, this problem could be sidestepped by a statistical analysis of the probability that two independent dies would have IDs that were close enough to be misidentified as a result of this noise. Unfortunately, the nature of cryptographic operations is such that not even a single bit can be incorrect. This will require a different approach to error correction.

Maes *et al.* describe a low-overhead approach to implementing a soft-decision helper algorithm [18]. They describe a method wherein one collects confidence data in each bit by taking between 10 and 100 measurements of the SRAM PUF with  $N$  output bits prior to provisioning. This yields an output estimate vector  $X \in \{0, 1\}^N$ , and a vector of error probabilities  $P_e$ , where the  $i$ th element of this vector corresponds to the probability that a measurement of  $X_i$  will be erroneous. Going forward,  $X$  serves as the “fuzzy secret,” and  $P_e$  is public information. It has been proven by Maes *et al.* that revealing  $P_e$  does not leak any min-entropy of the response  $X$ . This work goes on to describe an implementation where the above soft-decision helper algorithm is combined with Reed–Muller codes and a universal hash function to distill the PUF output bits to a full-entropy reproducible secret key. This implementation takes 1536 SRAM PUF response bits (78% min-entropy with an average bit-error probability of 15%), and distills these data down to a 128-bit full-entropy key with a failure rate of  $\leq 10^{-6}$ . This approach demonstrates the feasibility of using SRAM PUFs as cryptographic key sources in spite of the errors inherent in SRAM PUF output bits.

2) *Attacks on SRAM PUFs*: Because the SRAM PUF provides a secure key (as opposed to providing challenge–response functionality like the strong PUF), it relies on other conventional security primitives to keep that key protected while the chip is powered. As a result, any side channel or other vulnerabilities associated with the cryptographic hardware pose a threat to the secret key outputted by the SRAM PUF. In addition, since this key is kept secret, the modeling attacks used against strong PUFs cannot be used, since no input/output relations of the PUF should ever be revealed.

However, there are other ways identified in the literature to attack a SRAM PUF more directly. Many of these depend on the level of access that one has to the SRAM. If one can insert a “write” command, then one could leverage the NBTI to deliberately force individual bits toward “1.” If one could modify the temperature, one could potentially cause the PUF to fail by running the PUF outside its design

area. Finally, the ability for a SRAM cell to maintain its state depends on the supply voltage. If, during the turn-on process, the supply voltage is held for some time at a low ( $\sim 100$  mV) voltage, the thermal noise will induce a transition into the cell’s favored state, resulting in higher stability. However, if the voltage turn-on is fast, then cells become less stable. An attacker with access to the power channel could potentially control the stability of some of the SRAM PUF bits through this mechanism [12].

Recently, it has also been identified by Helfmeier *et al.* that the SRAM power-on state can be observed via near-infrared imaging of the SRAM during the turn-on transient. Once the SRAM “fingerprint” has been measured (the PUF response bits have been stolen), one can use focused ion beam (FIB) techniques to modify a second IC to have a matching fingerprint as the first by cutting traces and/or demolishing transistors in the SRAM cell [10].

Finally, one notes that SRAM data are not erased immediately on power down. The data remain “stored” in the SRAM cell for a certain short time after the cell is powered down due to an effect called “data remanence.” Oren *et al.* have demonstrated that this effect can be used to inject faults into the SRAM PUF. In doing so, one can noninvasively learn the SRAM PUF output bits indirectly [28].

#### D. Ring-Oscillator PUF Implementation

Yu and Devadas designed a delay-based weak PUF based on the ring-oscillator architecture, and proposed the first PUF key generation architecture that does not require traditional error correction [40]–[42]. The proposed index-based syndrome coding method is a departure from prior error-correction schemes based on code-offset syndrome [5], where the syndrome format enables soft-decision functionality without the complexities associated with an explicit traditional soft-decision error-correction decoder, which in general has a higher complexity than an equivalent hard-decision error-correction decoder.

In this architecture, several oscillator PUF banks are instantiated, with each oscillator bank comprising  $2k$  ring oscillators. A  $k$ -bit challenge is applied to each bank, to determine which oscillators correspond to the top delays, and which oscillators correspond to the bottom delays. The top and bottom rows are summed to produce  $x$  and  $y$ , respectively. These values are used to produce a single bit PUF output and associated “soft-decision” information corresponding to a PUF challenge. Specifically, the output bit is the sign of  $x - y$ . The “confidence” (discussed more in Section VI-D1) is derived from the magnitude of  $x - y$ .

Fig. 6 shows a simplified diagram for illustrative purposes. More complex “recombination” functions using xors or amplitude modulation based on additional challenge bits were used in actual implementation.

Each of the oscillators is configured with “challenge bits.” For the purpose of cryptographic key generation, these bits are fixed (see the “fixed challenge” in Fig. 6) in order to reproduce the same key each time.

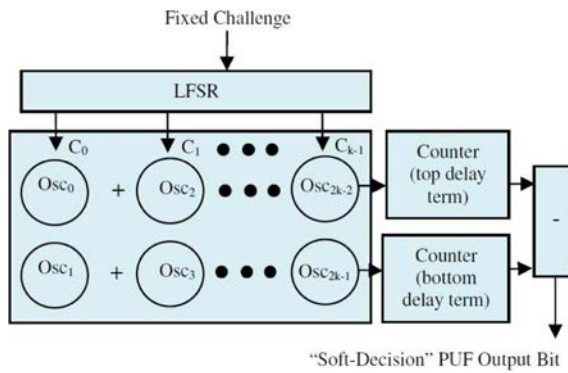


**Fig. 6. A  $k$ -sum ring-oscillator PUF. Ring oscillators that are closer in frequency do not affect the output bit due to the summation process.**

Yu *et al.* designed multiple circuit topologies to implement this technique using a  $0.13\text{-}\mu\text{m}$  CMOS process [42]. They found that, as expected, these devices produced output bits that passed all National Institute of Standards and Technology (NIST) standard randomness tests, demonstrated close to ideal decorrelation between different PUF devices, had a worst case bit bias less than 0.5%, and a raw intradevice variation of  $\sim 10\%$ .

1) *Ring-Oscillator PUF Error Correction:* Like the SRAM PUF, the ring-oscillator PUF also must correct for noise and environmental factors. The primary noise sources for ring-oscillator PUF architectures are similar to those for analog electronics. First, the die temperature and source voltage affect the gate delay in a known way. If the two frequencies are too close together, this noise will potentially change the output measurement bit.

The ring-oscillator PUF is inherently a differential measurement (measuring the difference in two identical sets of oscillators' paths). However, it is still susceptible to noise. As mentioned for arbiter PUFs, one can use block error codes, but this leaks some of the bits of the secret key. This is very undesirable for a weak PUF. In the case of the SRAM PUF mentioned above, Maes *et al.* were able to leverage the "confidence" in a given bit to improve coding efficiency. Yu and Devadas proposed a new error-correction scheme, "index-based coding" (IBS), that leverages this same principle for ring-oscillator PUFs [40] but does not require repeated PUF response measurements.

As mentioned above, if two sets of oscillators have frequency sums that are close together, then the readout circuit will not always report the same output bit due to slight frequency variations caused by noise and other environmental factors. Simply put, the output bit will take a value of either 0 or 1 with some probability. In such a situation, the "confidence" of a 0 or 1 measurement should be relatively low.

The IBS coding scheme recognizes the relative confidence in each bit measurement and adaptively chooses more

confident bits to be part of the PUF output. In doing so, this increases the response reproducibility and decreases error.

In addition to being more efficient than block codes, Yu and Devadas also showed that the scheme leaked no information about the PUF itself under the assumption that the PUF output bits are independent and identically distributed (i.i.d.). Note that i.i.d. is the same assumption used by Maes *et al.* in proving security of the use of the probability of bit error  $P_e$  in their scheme. This is a common assumption made about PUFs, but it is actually difficult to validate in practice. Nevertheless, this is to be contrasted with the previous use of block codes (Maes' method layers block coding on top of  $P_e$ , with  $P_e$  portion proved to be information theoretically secure), which do leak some information even under a PUF i.i.d. assumption (consider a PUF with a bias of 1% and use of repetition coding). By contrast, IBS, even when used with a second stage traditional (hard-decision) error correction, remains information theoretically secure under a PUF i.i.d. assumption, even for a heavily biased PUF. IBS decouples the PUF bias statistics from the syndrome leakage security in an i.i.d. PUF setting.

A full description of the IBS coding scheme is outside the scope of this paper. Yu and Devadas implemented an IBS code on a delay-based PUF with a challenge block size of 63 bits and an average of 25 errors to correct (35.9% error) in each block. Using the IBS approach, this was reduced to a 6-bit error (9.4%). It was identified that the probability of seven or more errors was less than 0.5 ppm, so a BCH (63, 30,  $t = 6$ ) code was used. This system successfully corrected errors across environmental conditions without failure.

In the study by Yu *et al.* in 2012, similar results were obtained [42]. In this study, and IBS coding scheme was used followed by the same BCH (63, 30,  $t = 6$ ) code. It was observed across the four extreme voltage-temperature corners that not only did the error-corrected PUF never output an erroneous bit, but also the maximum number of errors output from the IBS scheme (the number of errors that the BCH code had to correct) was three. The BCH code can correct up to six errors, so the design was shown to have a "stability safety margin" of 50%.

2) *Attacks on Ring-Oscillator PUFs:* Like the SRAM PUF, the ring-oscillator PUF relies on downstream cryptographic hardware/software to protect the security of the key that is generated. However, there are ways of potentially modifying the ring-oscillator PUF's behavior. Such an attack does not reveal the output key, but may be able to influence the device to either fail to regenerate a key (denial of service), or even manipulate secret key bits if such an attack were to occur during provisioning.

For example, it was shown in 2009 that driving a sinusoidal signal on the ground plane of a ring oscillator can cause it to "lock" to that signal [26]. This study demonstrated such an attack compromising a true random number generator (TRNG). Although an attack on a PUF

would have to take into account its differential nature, the same principle can be used. By locking the frequency, an attacker can drive the frequency of a given PUF to a desired value without invasive measures.

In addition, it was shown in 2011 that the electromagnetic radiation from the ring-oscillator PUF could also be used to steal the output bits [27]. This attack can be defeated by running several oscillators in parallel, which has been done in many studies on the ring-oscillator PUF, some of which predate the identification of the attack [40]–[42].

## VII. EMERGING PUF CONCEPTS

Although existing PUF technology has been successful in addressing applications in low-cost authentication and secure key generation, PUF technology still has significant untapped potential. New PUF architectures and applications are continually being developed. A full review of each of these paths is outside the scope of this paper, but a few of the popular emerging trends will be covered. For a more comprehensive coverage, we refer interested readers to a recent article on the topic [32].

### A. Model-based PUFs

When considering the application of low-cost authentication, one of the primary drawbacks of strong PUF architectures is the establishment of the secret challenge–response table. Not only does this require the server to securely communicate with the PUF prior to any authentication rounds in a “secure bootstrapping” phase, but also once a CRP is used, it must be discarded and never used again. Therefore, the server must collect a large number of CRPs at manufacture and store them secretly. For large applications with thousands of PUF clients, this corresponds to a large amount of required secret storage.

To mitigate these challenges, researchers have recognized that if a PUF could have an associated “secret model” that emulates the PUF challenge–response behavior, then the secure storage requirements could be alleviated [3], [21], [24]. The secret model in the case of an arbiter PUF would be the delays of the individual stages. Newer delay-based PUF constructions have even used this compact model to link software-based attestation to intrinsic device characteristics [14]. This linking enables secure timed (and even) remote attestation.

Such a “secret model” PUF still requires both the “secure bootstrapping” phase as well as the secure storage, as the PUF and authenticating server must “agree” on a secret PUF model that describes the PUF behavior. This model must be kept secret, as it exactly describes the PUF behavior and can be used to spoof an authentication sequence. However, a server may now choose any random challenge and independently compute the correct PUF response. Further, an encrypted model can be stored on the PUF device and a reader that knows the encryption key can authenticate the device in an offline fashion.

### B. Timed Authentication and Public Models

Although the secure model PUF architecture described above mitigates the secure storage requirement for PUF usage in authentication applications, it still requires secure bootstrapping and secure storage of the secret model.

Both of these requirements are alleviated by a new type of PUF described as timed authentication PUFs, public model PUFs (PPUFs), or SIMulation Possible but Laborious (SIMPL) systems [1], [20], [24], [33]. This paper will refer to this concept as a PPUF. An FPGA implementation of the PPUF was proposed alongside the concepts of an FPGA erasable PUF [20], [21]. A full characterization and compaction of the physical delays of the FPGA components is performed.

A PPUF has a model that emulates the challenge–response behavior of the PPUF hardware. This model is public—known to everyone. The key difference between the PPUF model and the PPUF hardware is that the PPUF hardware computes the response in a measurably faster time. Therefore, the authentication scheme works as follows (where a server is authenticating a Client):

- 1) server obtains the desired PPUF model from a trusted third party storage;
- 2) server generates a challenge and computes the response using the PPUF model;
- 3) server sends challenge to the client and begins timer;
- 4) the client uses its PPUF hardware to compute a response and sends it back to the server;
- 5) server measures the client response time  $T$ ;
- 6) server accepts if  $T < T_0$  and client’s response is correct.

In the above scheme, first note that the PPUF model is stored publicly. Although it may be publicly read, the PPUF model storage must be resistant against tampering or rewriting, as the server must be able to trust that a given PPUF model is associated with a certain PPUF hardware owner. This can be done using the traditional public key infrastructure (PKI) or other similar roots of trust.

In addition, the server must be able to establish some value  $T_0$  as described in the above scheme. This time is 1) long enough to allow the PPUF hardware to compute the response and allow for roundtrip network latency; and simultaneously, 2) short enough that no model could emulate the PPUF hardware and correctly produce a response in that time. This establishment of  $T_0$  is the fundamental challenge of designing a PPUF capable of enacting the above authentication scheme.

It is clear that such a PPUF system would have widespread application. The key recognition demonstrating the power of a PPUF is that the PPUF hardware contains no secrets. Counterintuitively, the device is still capable of securely authenticating itself to any server. The server also contains no secret information. Simply put, there is no secret information anywhere in the protocol. The authentication capability derives solely from the computational

difference between the hardware and the model, and the unclonability of the hardware.

With this in mind, applications in embedded security are immediately obvious. A modern embedded electronic device being authenticated by a server currently uses secure nonvolatile memory (NVM) to store secret bits. Even strong and weak PUFs can be considered as “storage” devices using manufacturing variation in variables such as dopant concentration to store secret bits. In all modern electronics, this secure storage acts as the “root of trust” on which all authentication and cryptographic mechanisms are based. If the secret bits can be stolen (in the case of secure NVM) or approximated (in the case of PUF modeling), then the security is broken.

In the case of a PPUF, there are no bits to steal. The security is instead based on the difficulty of reproducing an exact copy of the PPUF hardware. This represents a fundamental shift in security paradigms. Using this mechanism, a secure embedded system can be deployed in a highly untrusted environment with a strong threat model (an adversary already has access to both the PPUF design and PPUF model) and still act as a trusted authentication source.

### C. New PUF Architectures

The current primary open problem to PPUF design is the identification of a system with a provable  $T_0$  parameter as described in Section VII-B. Intuitively, one immediately would hope for a provable asymptotic separation between the PPUF computational hardware and the computational hardware used to execute the model. Such a separation has been observed between computers leveraging quantum effects and classical computers, but such quantum computational devices are still not close to the scale required for such cryptographic applications.

Therefore, classical systems must be considered for potential practical PPUF implementations. It is recognized that such an asymptotic speedup between the PPUF hardware and the computer running the model is not possible. Classical dynamical systems at a fundamental level are governed by local differential equations (e.g., Maxwell’s equations, Lagrangian mechanics, and Newtonian gravitation). As a result, one can see that a discretized universe can be simulated with only constant factor slowdown.

Qualitatively speaking, one can imagine a computer with a processor dedicated to simulating each point in

space and communicates with its neighbors. Each time step can then be simulated in constant time by this set of processors running in parallel. Therefore, with enough parallelism, any classical system can be simulated with only constant factor slowdown.

Therefore, if one accepts that the PPUF model will only be slower than the PPUF hardware by a constant factor, the next step is to design a system with bounds on this constant factor. If a model is provably  $10^6 \times$  slower than the PPUF hardware, then this constant factor is large enough to derive an acceptable  $T_0$  for PPUF operation.

Many PPUF architectures have been proposed [1], [20], [21], [33]. However, to date, the authors are unaware of a proposed PPUF architecture where such constant factor bounds are provable, or even those with bounds that can be strongly argued.

To establish such bounds, one first recognizes that any computational model will use CMOS technology, since CMOS is simultaneously the fastest and least expensive computational platform currently available. In recognizing this fact, one can then identify the minimum timescale of active CMOS devices as a comparison benchmark to the timescale of the differential equations describing the PPUF hardware system.

One potential avenue of approach that has been identified is in the use of optoelectronics. Optoelectronic systems simultaneously have fast enough internal dynamics to allow for significant constant factor slowdown, and are also integrable into existing CMOS processes.

In conclusion, this paper has introduced two primary applications of PUF technology: low-cost authentication and secret key generation. It has covered several of the most popular approaches to each of these applications, including arbiter PUFs, SRAM PUFs, and ring-oscillator PUFs. It has discussed potential mathematical and physical attacks on each PUF technology as well as popular error-correcting techniques for each. Finally, this paper discusses new PUF technologies such as PPUFs that demonstrate that PUF technology still has tremendous untapped potential. PUFs provide a new, secure technology for authentication and secure key storage with many advantages over existing approaches. New PUF error-correction approaches and technologies such as PPUFs represent an exciting new frontier for both PUF research as well as cryptography as a whole. ■

### REFERENCES

- [1] N. Beckmann and M. Potkonjak, “Hardware-based public-key cryptography with public physically unclonable functions,” *Information Hiding*, vol. 5806. Berlin, Germany: Springer-Verlag, 2009, pp. 206–220, ser. Lecture Notes in Computer Science.
- [2] C. Dainty, *Laser Speckle and Related Phenomena*. New York, NY, USA: Springer-Verlag, 1984.
- [3] S. Devadas, “Non-networked RFID PUF authentication,” U.S. Patent 8 683 210, U.S. Patent Appl. 12/623 045, 2008.
- [4] S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal, “Design and implementation of PUF-Based ‘unclonable’ RFID ICs for anti-counterfeiting and security applications,” in *Proc. IEEE Int. Conf. RFID*, May 2008, pp. 58–64.
- [5] Y. Dodis, L. Reyzin, and A. Smith, “Fuzzy extractors: How to generate strong keys from biometrics and other noisy data,” *Advances in Cryptology—Eurocrypt 2004*, vol. 3027. Germany: Springer-Verlag, 2004, pp. 523–540, ser. Lecture Notes in Computer Science.
- [6] B. Gassend, “Physical random functions,” M.S. thesis, Dept. Electr. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Jan. 2003.
- [7] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, “Silicon physical random functions,” in *Proc. 9th ACM Conf. Comput. Commun. Security (CCS)*, 2002, pp. 148–160.
- [8] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, “Identification and authentication of integrated circuits,” *Concurrency Comput., Practice Exp.*, vol. 16, no. 11, pp. 1077–1098, 2004.

- [9] S. Graybeal and P. McFate, "Getting out of the STARTing block," *Sci. Amer.*, vol. 261, no. 6, 1989.
- [10] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning physically unclonable functions," in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2013, DOI: 10.1109/HST.2013.6581556.
- [11] D. Holcomb, W. Burleson, and K. Fu, "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," presented at *Conf. RFID Security*, Malaga, Spain, Jul. 11–13, 2007.
- [12] D. Holcomb, W. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.
- [13] D. Kirovski, "Anti-counterfeiting: Mixing the physical and the digital world," *Towards Hardware-Intrinsic Security*, A.-R. Sadeghi and D. Naccache, Eds. New York, NY, USA: Springer-Verlag, 2010, pp. 223–233.
- [14] J. Kong, F. Koushanfar, P. K. Pendyala, A.-R. Sadeghi, and C. Wachsmann, "PUFatt: Embedded platform attestation based on novel processor-based PUFs," presented at the *ACM/IEEE Design Autom. Conf.*, San Francisco, CA, USA, Jun. 1–4, 2014.
- [15] P. Layman, S. Chaudhry, J. Norman, and J. Thomson, "Electronic fingerprinting of semiconductor integrated circuits," U.S. Patent 6 738 294, Sep. 2002.
- [16] J.-W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits with identification and authentication applications," in *Proc. IEEE VLSI Circuits Symp.*, 2004, pp. 176–179.
- [17] D. Lim, "Extracting secret keys from integrated circuits," M.S. thesis, Dept. Electr. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, May 2004.
- [18] R. Maes, P. Tuyls, and I. Verbauwhede, "Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs," *Cryptographic Hardware and Embedded Systems—CHES 2009*, vol. 5747. Berlin, Germany: Springer-Verlag, 2009, pp. 332–347, ser. Lecture Notes in Computer Science.
- [19] A. Mahmoud, U. Rührmair, M. Majzoobi, and F. Koushanfar, "Combined modeling and side channel attacks on strong PUFs," *Rep. 2013/632*, 2013.
- [20] M. Majzoobi, A. Elnably, and F. Koushanfar, "FPGA time-bounded unclonable authentication," *Information Hiding*, vol. 6387. Berlin, Germany: Springer-Verlag, 2010, pp. 1–16, ser. Lecture Notes in Computer Science.
- [21] M. Majzoobi and F. Koushanfar, "Time-bounded authentication of FPGAs," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pt. 2, pp. 1123–1135, Sep. 2011.
- [22] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *Proc. ACM/IEEE Int. Conf. Comput.-Aided Design*, 2008, pp. 670–673.
- [23] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *Proc. IEEE Int. Test Conf.*, 2008, DOI: 10.1109/TEST.2008.4700636.
- [24] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 1, 2009, DOI: 10.1145/1502781.1502786.
- [25] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender PUF protocol: A lightweight, robust, secure authentication by substring matching," in *Proc. IEEE Symp. Security Privacy Workshops*, 2012, pp. 33–44.
- [26] A. T. Markettos and S. W. Moore, "The frequency injection attack on ring-oscillator-based true random number generators," in *Proc. Int. Workshop Cryptogr. Hardware Embedded Syst.*, 2009, pp. 317–331.
- [27] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Semi-invasive em attack on FPGA RO PUFs and countermeasures," in *Proc. Workshop Embedded Syst. Security*, 2011, pp. 2:1–2:9.
- [28] Y. Oren, A.-R. Sadeghi, and C. Wachsmann, "On the effectiveness of the remanence decay side-channel to clone memory-based PUFs," in *Proc. Int. Workshop Cryptogr. Hardware Embedded Syst.*, 2013, pp. 107–125.
- [29] R. S. Pappu, Z. Paral, and S. Devadas, "Reliable and efficient PUF-based key generation using pattern matching," in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2011, pp. 128–133.
- [30] R. S. Pappu, P. S. Ravikanth, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, pp. 2026–2030, 2002.
- [31] M. Rostami, M. Majzoobi, F. Koushanfar, D. Wallach, and S. Devadas, "Robust and reverse-engineering resilient PUF authentication and key-exchange by substring matching," *IEEE Trans. Emerging Topics Comput.*, 2014, DOI: 10.1109/TETC.2014.2300635.
- [32] M. Rostami, J. B. Wendt, M. Potkonjak, and F. Koushanfar, "Quo vadis, PUF?: Trends and challenges of emerging physical-disorder based security," in *Proc. Conf. Design Autom. Test Eur.*, 2014, article 352.
- [33] U. Rührmair, "SIMPL systems: On a public key variant of physical unclonable functions," International Association for Cryptologic Research, Tech. Rep., 2009.
- [34] U. Rührmair, S. Devadas, and F. Koushanfar, "Security based on physical unclonability and disorder," *Introduction to Hardware Security and Trust*, M. Tehranipoor and C. Wang, Eds. New York, NY, USA: Springer-Verlag, 2012, pp. 65–102.
- [35] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proc. 17th ACM Conf. Comput. Commun. Security*, 2010, pp. 237–249.
- [36] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, F. Koushanfar, and W. Burleson, "Power and timing side channels for PUFs and their efficient exploitation," *Rep. 2013/851*, 2013.
- [37] Y. Su, J. Holleman, and B. Otis, "A 1.6 pJ/bit 96 (percent) stable chip ID generating circuit using process variations," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2007, pp. 200–201.
- [38] G. E. Suh, "AEGIS: A single-chip secure processor," Ph.D. dissertation, Dept. Electr. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Aug. 2005.
- [39] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. ACM/IEEE Design Autom. Conf.*, 2007, pp. 9–14.
- [40] M.-D. M. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 48–65, Jan./Feb. 2010.
- [41] M.-D. M. Yu, D. M'Raihi, R. Sowell, and S. Devadas, "Lightweight and secure PUF key storage using limits of machine learning," in *Cryptographic Hardware and Embedded Systems—CHES 2011*, vol. 6917. Berlin, Germany: Springer-Verlag, 2011, pp. 358–373, ser. Lecture Notes in Computer Science.
- [42] M.-D. M. Yu, R. Sowell, A. Singh, D. M'Raihi, and S. Devadas, "Performance metrics and empirical results of a PUF cryptographic key generation ASIC," in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2012, pp. 108–115.

## ABOUT THE AUTHORS

**Charles Herder** received the B.S. degree in electrical engineering and computer science, the B.S. degree in physics, and the M.S. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA.

He has published in many fields including physical chemistry, single-photon detection, embedded cryptosystems, and power management. His research interests are in the physics of computation, cryptography, computer architecture, and computer security. His prior experience at Texas Instruments includes developing embedded systems authentication technology and serving as lead technical point-of-contact for the development of proprietary power management and authentication systems for several major developers including Dell, RIM, Cisco, and Phillips.



**Meng-Day (Mandel) Yu** received the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA.

He is a Technical Director at Verayo, San Jose, CA, USA. Before that, he was the Manager of R&D Engineering at TSI, a secure wireless radio startup. Before that, he was an ASIC Engineer and later a Systems Engineer at TeraLogic focused on signal processing and conditional access systems. His research interests include coding theory, signal processing, and computer security.



**Farinaz Koushanfar** received the B.S. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1998, the M.S. degree from the University of California Los Angeles (UCLA), Los Angeles, CA, USA, and the M.A. degree in statistics and the Ph.D. degree in electrical engineering from the University of California Berkeley, Berkeley, CA, USA in 2005.

She is an Associate Professor of Electrical and Computer Engineering (ECE) at Rice University, Houston, TX, USA. She is the Director of the Adaptive Computing and Embedded Systems (ACES) Laboratory.



**Srinivas Devadas** (Fellow, IEEE) received the M.S. and Ph.D. degrees from the University of California Berkeley, Berkeley, CA, USA, in 1986 and 1988, respectively.

He is the Webster Professor of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. He joined MIT in 1988 and served as the Associate Head of the Department of Electrical Engineering and Computer Science, with responsibility for Computer Science, from 2005 to 2011. His research interests include Computer-Aided Design, computer architecture, and computer security.



# Public Physical Unclonable Functions

*This paper surveys the time-bounded or public PUFs, including their design and security evaluation and the new protocols that they can support.*

By MIODRAG POTKONJAK, *Member IEEE*, AND VISHWA GOUDAR, *Student Member IEEE*

**ABSTRACT** | A physical unclonable function (PUF) is an integrated circuit (IC) that serves as a hardware security primitive due to its complexity and the unpredictability between its outputs and the applied inputs. PUFs have received a great deal of research interest and significant commercial activity. Public PUFs (PPUFs) address the crucial PUF limitation of being a secret-key technology. To some extent, the first generation of PPUFs are similar to SIMulation Possible, but Laborious (SIMPL) systems and one-time hardware pads, and employ the time gap between direct execution and simulation. The second PPUF generation employs both process variation and device aging which results in matched devices that are excessively difficult to replicate. The third generation leaves the analog domain and employs reconfigurability and device aging to produce digital PPUFs. We survey representative PPUF architectures, related public protocols and trusted information flows, and related testing issues. We conclude by identifying the most important, challenging, and open PPUF-related problems.

**KEYWORDS** | Cryptographic protocols; physical unclonable function (PUF); public PUF (PPUF)

## I. INTRODUCTION

The initial impetus for modern cryptography was provided by the first public-key primitives and protocols in 1976. Using mathematical and algorithmic mechanisms, several primitives (building blocks) and numerous protocols that target a variety of classic cryptographic tasks have been proposed, implemented, and widely used. The initial emphasis was on secure communication between computers that are placed in physically secured locations.

Classic cryptographic mechanisms and protocols are among the most surprising and elegant algorithms within

the wide spectrum of computer science tasks. Although their mathematical correctness is still not proved, it is widely considered that they are secure. However, it has also been demonstrated that classic cryptographic systems are easily compromised using side-channel techniques and physical attacks.

More recently, a new type of security primitive, the physical unclonable functions (PUFs), has attracted a great deal of attention. A PUF is a multiple-input-multiple-output function that has hard-to-predict dependency between the outputs and the inputs. While the initial proposal used an optical mesoscopic system for demonstration, the tremendous growth in interest in PUFs is due to its standard semiconductor integrated circuit (IC) implementation. The uniqueness of identical PUF design is provided by currently ubiquitous process variation. Several PUF architectures (e.g., arbiter based, ring oscillator, and SRAM) have been proposed, implemented, and analyzed. The initial security protocol was secret key in which one party collects a set of challenge-response pairs before releasing the PUF to another party. The authentication of the second party can now be done by the first party by issuing a challenge. Only the entity with the PUF can respond to an unknown challenge fast.

A PUF is a low-cost, small-area, and power-efficient security primitive that has good resiliency against side-channel and physical attacks. It has also been demonstrated that PUFs can be used for the creation of a variety of security protocols. However, PUFs remain a secret-key primitive which is an essential constraint for their use in many applications.

The introduction of public physical unclonable functions (PPUFs) was motivated by a need to create an ultrafast, ultralow-power public-key security primitive. The definition of a PPUF is a multiple-input-multiple-output system that is much faster to execute than it is to simulate, and whose security no longer relies on the secrecy of its physical parameters as PUFs do. Instead, PPUFs derive their security from large, preferably exponential, discrepancies between the time required for

Manuscript received August 26, 2013; revised June 9, 2014; accepted June 14, 2014. Date of publication July 15, 2014; date of current version July 18, 2014. This work was supported in part by the National Science Foundation (NSF) under Awards CNS-0958369, CNS-1059435, and CCF-0926127. The authors are with the Computer Science Department, University of California Los Angeles, Los Angeles, CA 90095 USA (e-mail: miodrag@cs.ucla.edu).

Digital Object Identifier: 10.1109/JPROC.2014.2331553

0018-9219 © 2014 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See [http://www.ieee.org/publications\\_standards/publications/rights/index.html](http://www.ieee.org/publications_standards/publications/rights/index.html) for more information.

response calculation from the private key and the time required to simulate the correct response from the public key. This enables PPUFs to underlie several public-key protocols and provide all the novel facilities that this entails, including easier key management and distribution, document signing, data exchange with untrusted parties, etc. Further, PPUFs can outperform classical public-key security primitives in terms of delay, energy, and throughput requirements owing to simpler and computationally less expensive design, and improve upon their resilience to side-channel and physical attacks [1].

Our goal is not just to present the use of PPUFs in a wide spectrum of security protocols and applications ranging from cryptography to trust and privacy, but also to describe their use in other applications such as intellectual property protection of software and hardware, synthesis of secure architectures and systems, and other emerging applications. We emphasize three dimensions: security primitives, protocols, and underlining security and design paradigms.

## II. PPUF ARCHITECTURES

### A. Public Physical Unclonable Function

The first PPUF design used a simple structure of XOR gates which had an expected exponential amount of glitching relative to the depth of circuitry [2]. Fig. 1 illustrates PPUF operation with an example of a simple PPUF circuit and the variation in gate delay characteristics caused by manufacturing processes variation. Assuming a stable circuit output after the input vector “01” is presented, introducing a new input vector “10” at time  $t=0$  produces transient behavior: At  $t=8.8$  ps ( $t=9.3$  ps) the output at gate B (A) switches to “0”, and then to “1” at  $t=11.2$  ps ( $t=10.1$  ps). Similar transient transitions occur at each row of the circuit, with number of such transitions doubling at each row and ultimately producing an exponential amount of glitching at the circuit’s output relative to its depth.

More generally, a PPUF was constructed as a rectangular array of gates,  $w$  gates wide and  $h$  gates deep. If the

expected number of output transitions per input transition for each gate is  $B$ , the simulation cost of the PPUF was shown to be  $wB^h/2$ . Therefore, the PPUF structure exploits the gap between implementation and simulation, which is an exponential function of the logic depth of the circuitry.

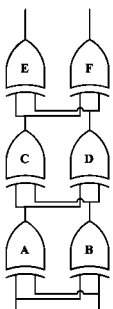
It is important to mention that there is another security primitive, SIMulation Possible, but Laborious (SIMPL), which also exploits the execution–simulation gap (ESG) [3]. Initially, the application range of SIMPL was restricted to authentication and required a relatively small ESG. However, Rührmair *et al.* have developed a variety of security protocols that exploit exponential ESGs [4]–[6]. Very recently, Horstmeyer *et al.* proposed hardware one-time pads that can serve as communication PUFs (cPUFs) [7]. Keys are extracted by optically probing random and unique optical structures using specially designed lasers. The keys are then shared using a theoretically perfect security procedure.

These first-generation PPUFs have several potential drawbacks including the need for ultra-accurate time measurements and detection of ultrashort glitches. While time can be measured with attosecond accuracy, the required instruments are very expensive. Furthermore, in order to achieve an acceptable advantage over attackers, both sides require computational efforts that are at least several seconds. Therefore, they are slower than the classical cryptography public-key cryptography protocols. These drawbacks were eliminated in more recent PPUF designs.

### B. Differential PPUF

The need for ultra-accurate timing and for detecting glitching was eliminated using a differential PPUF (dPPUF) [8]. Here, an input vector is simultaneously presented to two nominally equal circuits whose delay characteristics differ due to process variation. As the signals propagate through each circuit, they race against one another toward an arbiter which locks its output value to that of the first signal that arrives. Ultimately, it is the differential timing between the frontier signals of the two circuits that defines dPPUF operation rather than the absolute timing of a transient output vector in a PPUF circuit, and this eliminated the need for ultra-accurate timing.

The dPPUF design was presented in the context of an authentication protocol, wherein a verifier issues a challenge input vector to the user. The user, who is in physical possession of the dPPUF, executes the challenge to produce the response within a single clock cycle and presents this response to the verifier. To authenticate the user, the verifier compares this to a simulated dPPUF response to the challenge. Similar to a basic PPUF, the dPPUF-based protocol takes advantage of the ESG between the time it would take an attacker to simulate the response, and the time the user takes to execute the challenge on a dPPUF IC.



| Gate | Input 1<br>Delay (ps) | Input 2<br>Delay (ps) |
|------|-----------------------|-----------------------|
| A    | 9.3                   | 10.1                  |
| B    | 11.2                  | 8.8                   |
| C    | 11.1                  | 9.0                   |
| D    | 7.8                   | 7.1                   |
| E    | 8.6                   | 9.5                   |
| F    | 12.4                  | 9.6                   |

Fig. 1. Simple PPUF [2].

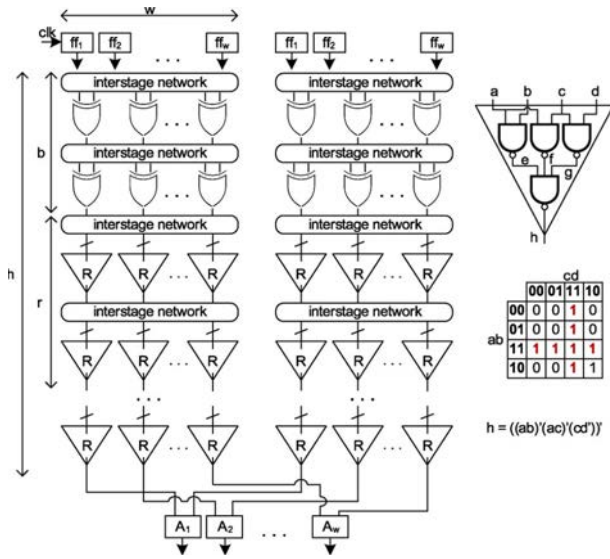


**Fig. 2.** dPPUF architecture of height  $h$  and width  $w$  [8]. A challenge vector propagates through repeated stages of  $b$  booster (xor) and  $r$  repressor (R) cells (right), with partly maximally interacting and partly random interstage networks. Signals from two nominally identical but physically unique PPUFs race to arbiters ( $A_i$ ) to produce the response.

Assuming  $p$  is the probability of the more likely output value (0 or 1) on each output line of a dPPUF, and  $q$  is the probability that a randomly chosen response will be successfully authenticated, the size of the output vector  $n$  required to achieve  $q$  is  $\log(q)/\log(p)$ . Therefore, to maximize the proof of strength ( $q$ ) on a dPPUF of given size, it is necessary to keep  $p$  as close to 0.5 as possible. In other words, output predictability dictates dPPUF's security.

Fig. 2 illustrates the dPPUF architecture proposed by Potkonjak *et al.* It uses the concepts of confusion and

diffusion to reduce predictability. It is composed of alternating layers of booster cells that amplify the extent of output switching, and repressor cells that unpredictably repress frontier signal transitions which would otherwise lock the arbiters. These concepts of repression and boosting are crucial to the creation of frontier signals that are highly unpredictable. Hence, when using a dPPUF, it is sufficient to detect a winner in a race between two signals using an arbiter.

Potkonjak *et al.* also studied the output predictability of the dPPUF architecture by measuring the output probabilities for each of its output bits over many input vectors [Fig. 3(a)]. The red dashed lines depict the ideal case where  $P(O_i = 1) = 0.5$  for all  $i$ . Similarly, the predictability for a single input vector across 1000 different dPPUF instances was also simulated [Fig. 3(b)]. The study concluded that the few suboptimal outputs should be excluded from verification in order to maximize dPPUF security. Finally, Fig. 3(c) illustrates the extent of correlation between output bits. If an attacker can successfully predict multiple outputs by simulating only a few, this affords him a proportional advantage in simulation time. Low correlations were observed between a large majority of output bit pairs.

Although dPPUFs eliminate many of the problems of the first-generation PPUF, they still require that at least one of the communicating parties invest significant resources and time in simulation.

### C. Matched PPUF

Matched PPUFs (mPPUFs) [9] are the first security primitive that enable the execution of a number of cryptographic and other protocols in a single clock cycle. The idea is simple and represents a complete departure from ESG: two identical PPUFs are created in such a way that creation of a third identical PPUF is essentially impossible. mPPUFs are created using both process

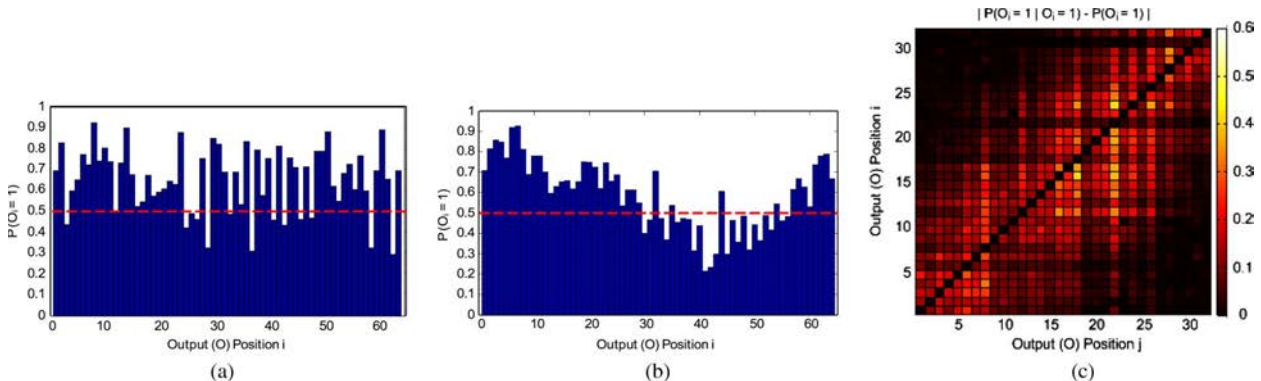


**Fig. 3.** For  $w = 64$  and  $h = 10$ , probability that an output bit will equal to 1 (a) for one dPPUF and (b) for 1000 different dPPUF instances given the same input. (c) Correlation between output bits  $O_i$  and other output bits  $O_j$  [8].

variation and device aging. Essentially, each party ages a large number of gates in their PPUFs in order to create identical paths through the circuit. Parts that cannot be matched are removed from the PPUFs using either power gating or software mechanisms. Since the PPUF delay characteristics are public, no storage or sharing of secret keys is necessary and the diverse applicability of public-key security may be availed. Further, by obviating the need for simulation, this security is achieved in a single-cycle, low-energy operation.

In [9], Meguerdichian and Potkonjak leveraged a controlled and reversible device aging technique (e.g., negative-bias temperature instability (NBTI)-induced transistor slowdown [10]) to age a subset of the gates of each communicating party and disable the unmatchable gates, thus producing perfectly matched PPUFs. Under the assumption that gate delays are randomly distributed between 0 and 1, and that device aging can increase a gate's delay by up to 0.5, Meguerdichian and Potkonjak showed that 25% of gates would not be matchable and an attacker making use of the publicly available delay characteristics to match his PPUF to that of the communicating parties would succeed in doing so for only 58% of his matched gates. On the other hand, the infeasibility of simulation-based attacks is apparent.

Fig. 4 illustrates the mPPUF architecture which is similar to the dPPUF architecture (Fig. 2). It is also similar to the dPPUF in that its output vector depends on the frontier signal which, in the case of an mPPUF, races against the clock. This architectural detail has been borrowed to enable a continued independence from ultra-accurate timing requirements. The mPPUF architecture also uses boosters and repressers toward minimizing predictability. While boosters increase switching frequency, the role of repressers is to decrease the switching frequency in an unpredictable manner in order to repress frontier signal transitions and thereby increase simulation complexity and unpredictability. For a repressor such as a  $k$ -input NAND gate, the unpredictability of its output

depends on the probability distribution of its input. As it turns out, the effect of multiple stages of identical repressers is a decrease in the randomness of input between consecutive repressor stages, in a manner that increases switching frequency in a more predictable manner rather than decreasing it in an unpredictable one. It was shown that alternating among different classes of repressers between stages mitigates this effect [9]. Consequently, the application of repressers was refined to the use of different classes of repressers at different stages. The final stage of the mPPUF architecture is a terminator cell, such as a  $k$ -input OR gate, that increases the stability of the inputs to the arbiters.

#### D. Quantized PPUF

Quantized PPUFs (qPPUFs) eliminated the last mPPUF bottleneck—the need that a user uses a different PPUF in communication with different parties [11]. Since mPPUFs require the delays of corresponding PPUF gates to be closely matched, the number of matchable gates reduces exponentially as the number of communicating parties increases. qPPUFs inherit their architectures from mPPUFs (Fig. 4). However, they differ in that each user's PPUF is aged independently of all other users. Matching then boils down to identifying and disabling those gates corresponding to the pair of communicating PPUFs which cannot be matched.

In order to accomplish matching with  $k$  different parties, a user disables  $k$  different sets of components on his PPUF. Note that he separately communicates with each of the other users, enabling and disabling the corresponding set of PPUF components in turn. Since aging is accomplished independently for each PPUF, switching communication to a different user in this manner can be accomplished in real time. Consequently, this new mPPUF enables a user to use a single PPUF to communicate with an arbitrary number of parties in which each of them requires only a single cycle computation.

qPPUFs achieve independent aging by quantizing the delay profile of each component class to a limited set of

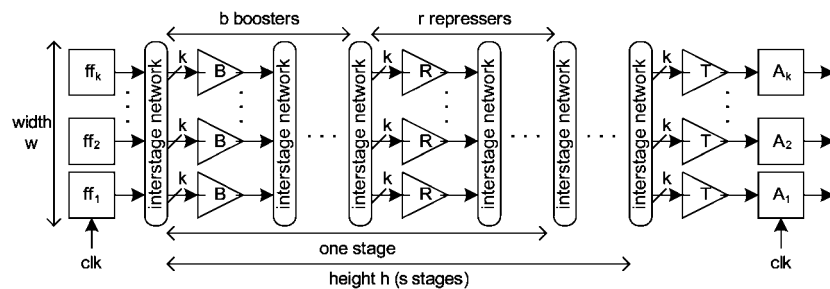


**Fig. 4.** mPPUF architecture of width  $w$  and height  $h$  [9], consisting of  $s$  stages of  $b$  booster cells ( $B$ ) and  $r$  repressor cells ( $R$ ), with a final level of terminator cells ( $T$ ) to enhance stability. Signals from input flip flops ( $ff$ ) race against the clock to arbiters ( $A$ ), which output 0 if the signal transitions before the clock and 1 otherwise.

values. A user then ages each component to coincide with an achievable quantum, thereby eliminating the need for coordinated aging. If a component cannot be aged to any of the preselected quanta, it is disabled. And if a component can be aged to multiple quanta, one is chosen randomly. Fig. 5 depicts such a quantization for a sample PPUF component with a Gaussian delay profile (mean = 1, standard deviation = 0.1).

To match their PPUFs, one user from each user pair transmits its delay quanta assignments for each component. This is followed by a reply message identifying the components that could not be matched and must be disabled for that user pair. An  $n$ -to- $n$  public-key communication can follow, wherein each of  $n$  users in a group achieves secure pairwise public-key communication with other members of the group, using a single PPUF while incurring little overhead when switching between communication partners, akin to the one described in [11].

A major drawback of quantization was identified to be a drastic reduction in matching probability. For the delay distribution in Fig. 5 and with two quanta, Meguerdichian and Potkonjak showed that the matching probability is about 0.35. This implies a similar reduction in the effective size and security of this qPPUF. While one solution is to increase the size of the PPUF, a more effective alternative was shown to be one where each PPUF component is replicated to increase its probability of getting matched.

Fig. 6 shows the change in matching probability as the number of replicas and the number of quanta are varied under the same delay distribution as in Fig. 5. Increasing the number of quanta improves the matching probability up to a point, after which the quantization

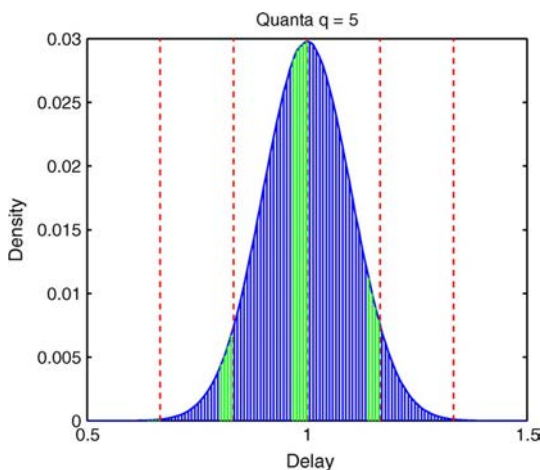


**Fig. 5.** qPPUF quantization example, showing possible quanta (red dashed) and gate delays that can be aged to match some quantum (shaded) for five quanta. Those gates with delay in the green shaded regions can age to two quanta (one is chosen at random) [11].

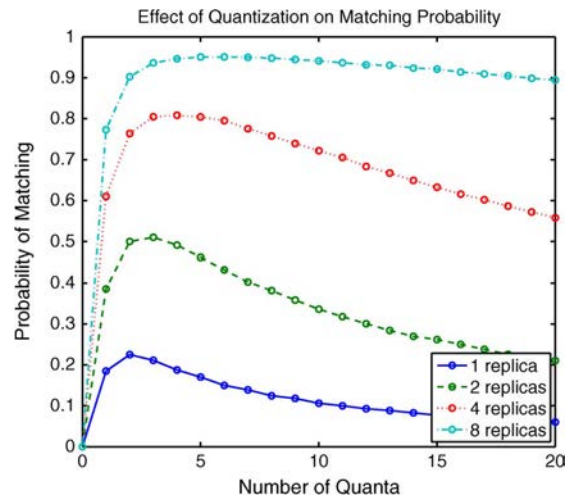


**Fig. 6.** Effect of varying quantization and component replication on matching probability [11].

intervals begin to overlap, thereby reducing the matching probability. This reduction arises from the randomized assignment of a delay quantum to a component when it can satisfy multiple quanta. In contrast, increasing the number of component replicas monotonically improves matching probability.

Finally, mathematical and statistical models of gate delay, threshold voltage, effective channel length, and aging were applied to study the behavior of qPPUFs [12]–[15]. Via simulation, Meguerdichian *et al.* also showed that the output predictability of a qPPUF is near-optimal for a vast majority of its output bits ( $w = 128$ ,  $b = 2$ ,  $r = 1$ ,  $s = 7$ , eight replicas, two quanta). Further studies showed that the probability that an attacker could match a single component of his PPUF to already matched components of two users was greater than 0.8 but less than 1. However, it was argued that with just 2000 PPUF components, the probability that an attacker fully matches the qPPUF was on the order of  $10^{-92}$ , despite a probability of matching a single PPUF component equal to 0.9.

### E. Digital PPUF

Digital PPUFs aim to provide low-energy public-key security that is independent of operational and environmental conditions, as well as, natural device aging [16]–[19]. This was achieved by avoiding an analog mechanism as the basis of the underlying security primitive. A pair of randomly generated digital bimodal functions (DBFs) is used instead. One of these functions  $f_{\text{compact}}$  can be computed quickly while its companion  $f_{\text{expand}}$  is unacceptably computationally intensive. In other words, digital PPUFs reverted to the use of an ESG to gain a disproportionate computational advantage over an attacker.

The following equation

$$\begin{aligned}
 &\text{Input: } a_i, i \in \{0, 1, 2 \dots 11\} & (1a) \\
 &\text{Output: } c_j, j \in \{0, 1, 2, 3\} & (1b) \\
 &\quad b_j, j \in \{0, 1, 2, 3\} & (1c) \\
 &\left\{ \begin{aligned} b_0 &= a_0 a_1 a_2 + \bar{a}_1 a_3 \\ b_1 &= a_4 \bar{a}_5 a_6 + \bar{a}_4 \bar{a}_7 + a_5 a_7 \\ b_2 &= a_8 \bar{a}_9 + a_9 \bar{a}_{10} \bar{a}_{11} \\ b_3 &= \bar{a}_0 \bar{a}_4 a_8 + a_0 \bar{a}_{10} \end{aligned} \right. & (1d) \\
 &\left\{ \begin{aligned} c_0 &= b_0 b_1 b_2 + \bar{b}_1 b_3 \\ c_1 &= b_1 \bar{b}_0 b_3 + \bar{b}_1 \bar{b}_3 + b_0 b_2 \\ c_2 &= b_1 \bar{b}_3 + b_2 b_0 \bar{b}_1 \\ c_3 &= \bar{b}_0 \bar{b}_2 b_3 + b_1 \bar{b}_3 \end{aligned} \right. & (1e) \\
 &\left\{ \begin{aligned} c_0 &= a_0 a_1 a_2 a_4 \bar{a}_5 a_6 a_8 \bar{a}_9 + a_0 a_1 a_2 a_5 a_7 a_8 \bar{a}_9 \\ &\quad + a_0 \bar{a}_{10} a_4 \bar{a}_5 \bar{a}_6 + a_0 a_1 \bar{a}_{10} \bar{a}_{11} a_2 a_9 + a_0 \bar{a}_{10} a_4 a_5 \bar{a}_7 \\ &\quad + a_0 a_1 a_2 \bar{a}_4 \bar{a}_7 a_8 \bar{a}_9 + a_0 \bar{a}_{10} \bar{a}_4 \bar{a}_5 a_7 + \bar{a}_0 \bar{a}_4 \bar{a}_5 a_7 a_8 \\ &\quad + \bar{a}_1 a_3 a_4 \bar{a}_5 a_6 a_8 \bar{a}_9 + \bar{a}_1 a_3 a_5 a_7 a_8 \bar{a}_9 \\ &\quad + \bar{a}_1 a_3 \bar{a}_4 \bar{a}_7 a_8 \bar{a}_9 + \bar{a}_1 \bar{a}_{10} \bar{a}_{11} a_3 a_4 \bar{a}_5 a_6 a_9 \\ &\quad + \bar{a}_1 \bar{a}_{10} \bar{a}_{11} a_3 a_5 a_7 a_9 + \bar{a}_1 \bar{a}_{10} \bar{a}_{11} a_3 \bar{a}_4 \bar{a}_7 a_9 \\ c_1 &= \dots \\ c_2 &= \dots \\ c_3 &= \dots \end{aligned} \right. & (1f)
 \end{aligned}$$

presents a sample digital PPUF where (1d) and (1e) comprise  $f_{\text{compact}}$ .  $f_{\text{expand}}$  is derived by substituting (1d) into (1e) and reducing to a minimal sum of products [see (1f)] or product of sums. Note that, in the general case, it is required that the number of inputs equals the number of output functions, i.e., the cardinality of  $i$  equals to that of  $j$ . Here, each set of subfunctions  $b_i$  is identical to the subfunctions  $c_i$ . In general, a digital PPUF can require several such iterations of the output subfunctions to arrive at the DBF  $f_{\text{expand}}$ . It is also required that the number of unique inputs to each output subfunction be bounded by some  $k$  to enable field-programmable gate array (FPGA) configuration and synthesis of the subfunctions in  $f_{\text{compact}}$ .

While  $f_{\text{compact}}$  and  $f_{\text{expand}}$  are equivalent, and  $f_{\text{expand}}$  can be derived from  $f_{\text{compact}}$  via iterative substitution and reduction, the inherent difficulty in functional decomposition makes it impossible to derive  $f_{\text{compact}}$  from  $f_{\text{expand}}$ . Further, Xu *et al.* [16] show that the number of product terms in  $f_{\text{expand}}$  exponentially increase with the number of primary inputs  $a_i$  and the number of iterations of the output subfunctions, whereas the number of terms in  $f_{\text{compact}}$  only increases linearly. Herein lies the ESG setting  $f_{\text{compact}}$  as the private key and  $f_{\text{expand}}$ , or some nontrivial subset thereof, as the public key makes it infeasible for an attacker armed with the public key to compute the DBF

output. Xu *et al.* [16] discussed this infeasibility and showed that with 20 iterations and ten inputs, the simulation time of  $f_{\text{expand}}$  is close to seven orders of magnitude slower than the execution time of  $f_{\text{compact}}$ . Based on these parameters, it was also shown that defeating the public-key communication scheme described below will take an attacker about 283 years.

A digital PPUF can be distributed and enlisted for public-key communication from A to B in the following manner. B selects a sizeable subset of  $f_{\text{expand}}$  and transmits it to A. Then, A chooses to store a random and manageable subset of the set of terms transmitted by the B as B's public key. Let the number of terms in this public key be  $l$ . For each such term  $f_i$ , A randomly selects its sum-of-products or product of-sums representation. Further, it selects a random term in the selected representation. Next, it generates the input vector  $p_i$  that would make  $f_i(p_i) = r_i = 1$  if a product term was selected, or  $f_i(p_i) = r_i = 0$  if a sum term was selected. Note that it is sufficient to select  $p_i$  as the input that would set the selected random term from  $f_i$  to 0 or 1, depending on the functional representation chosen, and this can be accomplished in linear time. Then, A concatenates each  $p_i$  for  $i \in \{1, \dots, l\}$  to construct a vector  $P_1$ . Similarly, vector  $R_1$  is generated from the  $r_i$ 's. This process is repeated  $N$  times to generate  $N$  vectors  $P_j$  and  $R_j$  for  $j \in \{1, \dots, N\}$ .  $N$  is a customizable parameter that further boosts the ESG and increases security while also increasing the energy costs and execution time.

Finally, A encrypts its message with the series of vectors  $R_j$  and broadcasts the encrypted message along with the vectors  $P_j$ . B, who is in possession of the private key  $f_{\text{compact}}$ , quickly computes vectors  $R_j$  from  $P_j$  and may then decrypt the message. An attacker overhearing the communication between A and B must, on average, compute half of the potentially exponentially number of terms transmitted by B to successfully decrypt the message, making this proposition infeasible.

Fig. 7 illustrates the sequential logic cluster (SLC) architecture that is designed to compute  $f_{\text{compact}}$  for a given input vector. It can also be used to derive the subset of  $f_{\text{expand}}$  that forms the basis of the public key. The input vector undergoes random shuffling before entering  $w$   $k$ -input lookup tables (LUTs) that each applies the combinational logic for the corresponding output subfunction of  $f_{\text{compact}}$ . The output of each iteration is stored in flip-flops at the end of computation before being fed back into the input stream for the next iteration. With this architecture, a large number of unrepeatable SLCs can be produced simply by altering the random shuffling or the contents of the SRAM cells in the LUTs.

The performance of digital PPUFs was analyzed in terms of the optimality of output prediction [Fig. 8(a)], and the extent to which the value of each output bit depends on that of the other output bits [Fig. 8(b)]. It was found that output prediction was near-optimal for all bits

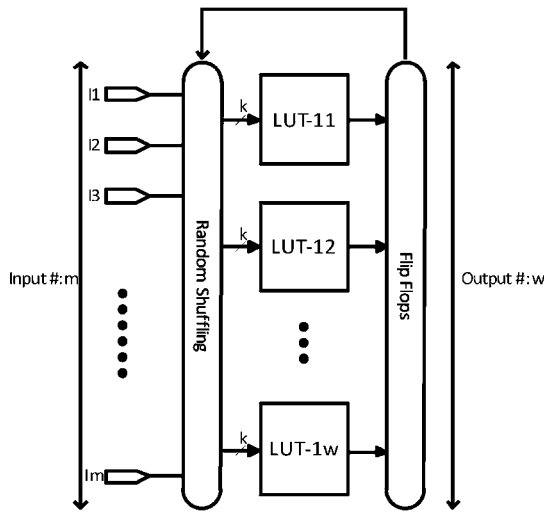


**Fig. 7. Architecture of a sequential logic cluster [16].**

in the average case, and that a majority of output bit pairs were near-independent (conditional probability close to 0.5). It was also shown that the energy costs of the SLC implementation-based decryption is on par with that of hardware implementations of AES and three orders of magnitude better than that of RSA. A major drawback is the overhead involved in the repeated communication of the large subset terms from  $f_{\text{expand}}$  that form the basis of the public key. However, technological trends show communication speeds outperforming storage speeds [20], which translates to a widening gap between requirement of an attacker who must store this large subset of terms, and that of the communication parties who must merely share them.

#### F. Nano-PPUF

Nano-PPUFs provide a faster, low-energy and more secure alternative to device aging, to overcome the burden of simulating the entire PPUF at one of the communicating parties (e.g., basic and differential PPUFs) [21]. Nanotechnology components are inherently faster and use lower power than current complementary metal–oxide–semiconductor (CMOS) technologies. Further, they naturally exhibit nonlinear circuit characteristics, in addition to the randomness that is inherent to their synthesis. This yields an ESG that is larger than in circuits composed of linear components. Nanotechnology components also express the unique characteristic of bidirectional signal propagation: Since input signals can be applied at any end of the circuit, this exponentially increases the input and output spaces with respect to a comparable CMOS PPUF. Together, these characteristics make nano-PPUFs more secure than conventional PPUFs.

Fig. 9 illustrates an example nano-PPUF. It is composed of a grid of nano-PPUF cells, with adjacent

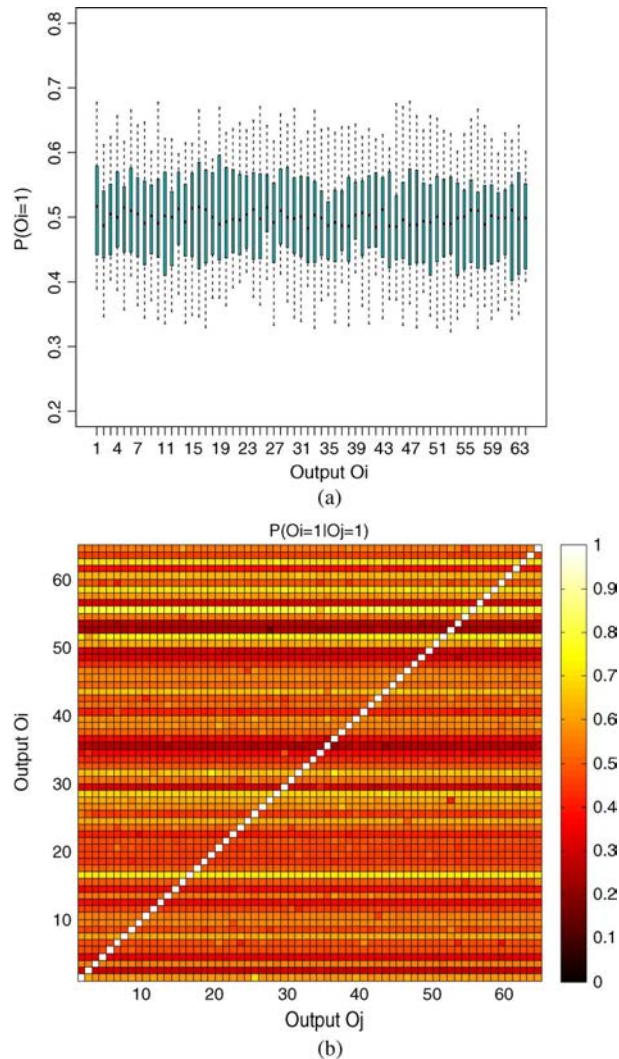


**Fig. 8. For 64 primary inputs, 32 iterations, and 1 000 000 input vectors: (a) probability that an output bit is equal to 1, and (b) conditional probabilities between output bits  $O_i$  and other output bits  $O_j$  [16].**

cells connected over matching pins. Each cell is modeled as a geometric random network generated during the synthesis of III–V nanowires. Nodes within the network are uniformly distributed and connected based on threshold distances. Each node exhibits nonlinear current–voltage characteristics, which gives rise to the nonlinear circuit characteristics of nano-PPUF cells and grids. The bidirectional signal propagation property of these circuits allows for inputs to be applied at any of the pins at the boundary of each cell. Therefore, nano-PPUF inputs are defined as a combination of input values (the voltages applied at the inputs) and input sets (the set of pins at which the voltages are applied). The rest of the boundary pins define nano-PPUF outputs, and are similarly composed of output values and output sets.

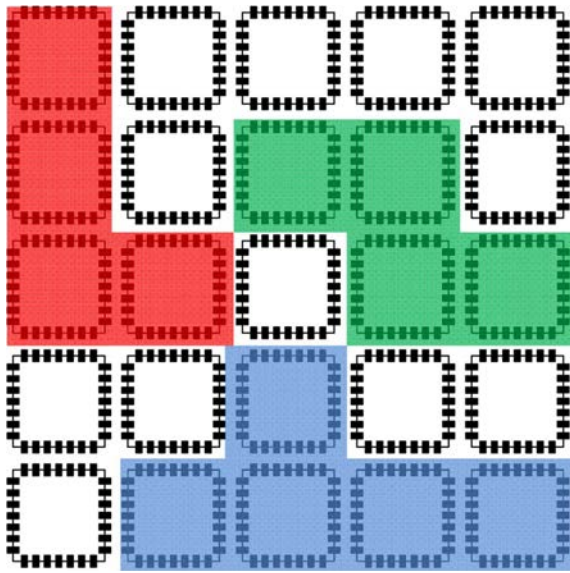


**Fig. 9.** Example of a  $5 \times 5$  nano-PPUF grid. Adjacent nano-PPUF cells are connected via adjacent pins. Example polyomino partitions of size 4, 4, and 5 are shaded [21].

To enable runtime definition of input and output sets, each pin at the boundary of a cell is multiplexed to facilitate input or output. This is what allows for arbitrary definitions of inputs and results in the exponential growth of the input/output spaces. Specifically, Wendt and Potkonjak [21] define a polyomino partitioning strategy where the set of cells accepting inputs are adjacent and connected, and formed by joining one or more equally sized squares edge to edge (see Fig. 9). It is shown that this

strategy ensures an input space that is exponential in the number of cells.

Two protocols are described as applications for nano-PPUFs: authentication and public-key communication. Authentication works as follows: The verifier issues a challenge to the user as an input to the user's nano-PPUF (input set and values). The user executes the challenge on his PPUF and presents the response to the verifier. The verifier simulates one or a few partitions of the user's nano-PPUF to verify that the input and outputs along their boundaries converge to the response provided. An attacker cannot know which partitions the verifier may simulate and must simulate the response over the entire nano-PPUF circuit, which is an infeasible task if the circuit is large enough.

Public-key communication is achieved as follows: The sender simulates a challenge and response on a partition of the receiver's nano-PPUF, and encrypts his message with the challenge. He transmits the encrypted message along with the response and the input set of the challenge to the receiver. The receiver searches for the input values corresponding to the challenge until the response values are matched. Once the entire challenge is recovered, the message is decrypted. On average, an attacker must simulate a search over half the set of all input values for the given input set, which is exponential in the size of the input set, making the attack infeasible.

Wendt and Potkonjak assessed the security of nano-PPUFs via SPICE circuit simulations. The study showed low correlation between input values and outputs. It was observed that even if such a correlation was discovered, given that the number of possible input sets is exponential in the size of the circuit, this would make it difficult to compute and store all sets of relationships for future lookup. In contrast, simulations showed strong correlations between a small set of output pairs (Fig. 10).

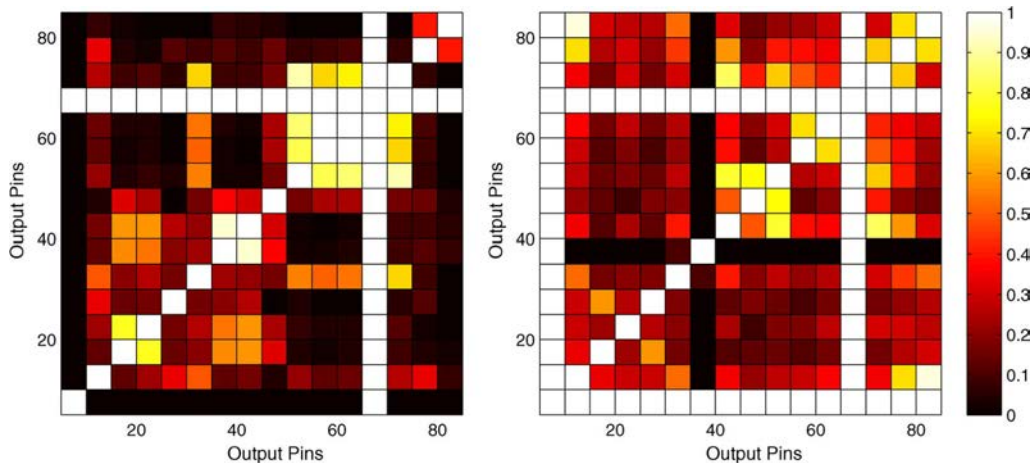


**Fig. 10.**  $R^2$  correlations between a subset of nano-PPUF output pins. Both figures depict the same nano-PPUF with different input sets. Pin 65 has perfect correlation because it returns a constant value due to process variation at network synthesis [21].

However, the correlated pairs changed when the input set changed. Consequently, Wendt and Potkonjak again posited that it would be infeasible to compute and store an exponential number of such correlations for each input set. Finally, simulation attacks were observed to be infeasible due to the exponential relationship between the circuit size and simulation time.

### III. PPUF PROTOCOLS

In this section, we sample from a wide spectrum of security, privacy, and trust protocols that employ PPUFs as a hardware primitive. Our emphasis is more on demonstrating new paradigms and strategies that facilitate the creation of protocols than on providing comprehensive coverage. The most important observation is that conceptually entirely new paradigms with respect to classical and quantum cryptography are required and created. What is common to all of them is that they are often surprisingly simple, and that all of them are intrinsically resilient against physical and side-channel attacks. For each of the representative protocols, we present their procedures when gap-based or matched PPUFs are used. Here, the most important observation is that matched PPUF-based protocols are invariably ultrafast and energy efficient: they rarely require more clock cycles and their PPUFs often have only a few hundred gates.

#### A. Public-Key Cryptography

First, we consider gap-based public-key communication protocols and follow that with a few modifications mainly related to the use of public-key infrastructure protocols presented by Beckmann and Potkonjak [2]. As with classical cryptography procedures, all PPUF-based public-key cryptography protocols assume the existence of a third trusted-party public-key infrastructure to prevent false impersonation attacks where the attacker pretends that he is one of the legitimate parties. Let us consider the situation where party *A* wants to send a secret message to party *B* using a gap-based scheme. Both parties *A* and *B* are in possession of their PPUFs whose public characteristics have been deposited with the trusted party *T*. Recall that for a given input *I* anybody can use simulation and calculate the correct corresponding output *O* using long simulations that, at the very least, require times in range of seconds. At the same time, the execution of *A*'s or *B*'s PPUF can be accomplished in the nanosecond range or faster.

The key challenge is to now create protocols that leverage this gap between the execution and simulation times, say of a factor of  $10^{12}$ , into two advantages for each of participating sides over the potential attacker. The advantage of side *A* that has to decrypt the message from *B* is obvious and provided directly by the ESG. The advantage of side *B* arises in the following manner. *B* selects a large set of numbers *S* that has on the order of  $10^{12}$  elements, and sends it to *A*. Set *S* has a compact representation. For example, it may be  $10^{12}$  numbers starting from a certain

specified number. *B* now randomly selects an element from *S* and uses it as input *I* in a simulation to calculate the corresponding output *O* of *A*'s PPUF. Finally, it uses bitwise XOR of the input *I* with the actual message *AM* to create the encrypted message *EM* and sends it to *A* together with the output *O*.

Now *A* uses its PUF to find the input that corresponds to output *O*, trying all inputs from set *S*. Once that input is identified, all that is required is for it to XOR input *I* and encrypted message *EM* to obtain the actual message *AM*. *A* can do it in a few seconds because it has its fast PPUF. Recall that *A* has a  $10^{12}$  factor speed advantage over the attacker.

*B* also has similar advantage because it has to compute only single challenge–response pair as it is the one that selects the input *I*. We see that, by selecting the size of set *S*, *A*, and *B* can optimize their relative execution times. There are other degrees of freedom that can be used for boosting this advantage over the attacker. For example, each of the legitimate parties may have multiple PPUFs where message sender randomly selects which one is used. Modern ICs have billions of gates, and each PPUF requires no more than a thousand gates.

Note that the presented protocol ensures that the attacker cannot decrypt the encrypted message, but, in this form, it allows impersonation attacks. However, a very small alteration is sufficient to prevent this attack. For instance, *A* can send a message *M* to *B* that is encrypted using *B*'s PPUF and is used for additional XOR in the protocol presented above. This message can only be decrypted using *B*'s PPUF in reasonable time. Finally, we conclude that it may be inferred from the ESG of the presented protocol that the expected time to crack the encryption is more than 100 000 years, and may be further boosted in several ways.

We now consider public-key cryptography for matched PPUFs. For the sake of brevity, we discuss quantized PPUFs. Again, we assume that there is a third trusted-party public-key infrastructure where all parties deposit the delays of each gate in the standard PPUF structure, after all the gates have been aged to the standardized quantization level. If two parties want to communicate with each other, they disable all unmatched gates in their PPUFs and begin their communication. The expected cardinality of the two matched PPUFs is well beyond the feasibility of simulation in any reasonable time on any reasonably sized distributed computing platform.

Now, there are several ways how public-key communication can be organized. In the simplest instance, all that is required is for party *B* to select an input *I* and calculate the corresponding output using the PPUF configuration that is matched to that of *A*. The final step is XORing of the output *O* and actual message *AM* into the transmitted message *SM* that is sent to party *A* together with input *I*. Party *A* also required only two steps. The first is the calculation of output *O* and the second is XORing of the output *O* and *SM* which yields the actual message *AM*. Note

that, in this protocol, the exchange of only a single message which is composed of input  $I$  and the encrypted message  $SM$  is sufficient. The only required computations consist of one PPUF calculation and a single bitwise XORing. Hence, this is the fastest and lowest energy realization of a public-key protocol.

### B. Secure Location Authentication

Secure location authentication enables verifier  $V$  to provably establish the claimed location of another participating party  $A$  [22], [23]. Recall that the verifier can deploy  $n$  measurement devices and ask  $A$  to send signals to each of them. In 3-D space, it is sufficient to use four measurement devices. The problem has not been solved in the domain of classical cryptography. A scheme where  $V$  asks party  $A$  to send its digital signature for a specified challenge along with the global positioning system (GPS) signal is not effective because  $A$  can provide its secret key to another helping party that serves as its proxy. Interestingly, the secure location problem can be solved using quantum cryptography.

The problem is also easily solved using any of the proposed PPUF designs. And at least in the case of matched PPUFs, its energy expense and time requirements are minimal and correspond to a few computational cycles if only digital components are considered. The idea is to slightly modify the procedure for remote trusted sensing and/or trusted computing. Note that, in these protocols, the verifier's PPUF can load any data to its PPUF, while  $A$  can receive its GPS data only through an antenna.

The essential observation is that  $A$ 's PPUF cannot be replicated and, therefore, only  $A$  can correctly respond to  $V$ 's challenges.  $A$  cannot remotely help its proxy because this would introduce additional communication delays. In addition,  $A$  would have to produce PUF results for challenges and the GPS signals, and accurate results are possible only if  $A$  is actually at the claimed location or if its PPUF can use data from memory. The latter case would require hardware changes that would invalidate the delays of  $A$ 's PPUF.

### C. Matched PPUF-Based $k$ -Anonymity Security Protocol

Privacy has emerged as one of the dominant system constraints and essential application desiderata for numerous important domains. For example, it is particularly important to mobile users (e.g., cybercar applications that provide complete information about locations of cars and their trajectories), and in the financial, medical, and social network domains which may provide sensitive information about users. There have been a number of attempts to solve this problem, for instance, zero knowledge techniques have been proposed to preserve user privacy of authentication at toll booths. While these techniques appear algorithmically sound, their computational requirements are excessive: even the most efficient implementations

require days for execution on modern personal computers (PCs). Gap PPUF-based solutions are somewhat faster but still require computational efforts with runtimes in the range of seconds. However, quantized matched PPUFs produce protocols that require only a few clock cycles, i.e., nanoseconds or so.

The essence of the approach is as follows. Let us assume that a toll owner (verifier) has  $m$  drivers (users), each equipped with a PPUF with uniform architecture. The PPUFs follow the quantized matched PPUF paradigm. Therefore, the verifier can configure her PPUF in such a way that it has identical input/output relationship with any subset of users. The larger the required overlap of matched gates, the smaller the cardinality of this subset will be. At the extreme, if the necessary overlay is very high, only a single user matches the service provider.

The basic  $k$ -anonymity protocol starts with a step where the verifier partitions all users into  $n$  groups so that each group has  $k_1$  drivers. For each group, she finds all gates that are shared among all  $k_1$  drivers in that group. At the first interaction step, she sends the characteristics of each of the  $n$  quantized matched PPUFs to the corresponding drivers, together with a randomly generated challenge.

If a driver is able to calculate the correct response to the challenge corresponding to one of the quantized matched PPUF characteristics, it is a potential sign that he indeed is one of paying subscribers. The chances that an attacker can compute the correct answer can be calculated using the following reasoning. If we denote the probability of matching two corresponding gates by  $p_g$  and the number of gates in each PPUF by  $n_g$ , then the probability  $p_{a1}$  that the attacker matches with a particular group is equal to  $p_g^{n_g}$ . Hence, the attacker's chance to match with any one of the  $n$  groups is equal to  $1 - (1 - p_{a1})^n$  which, for realistic numbers, is a very small quantity.

This probability can be reduced using the following procedure. The verifier additionally partitions users in the group for which she received the correct responses into two groups so that each group contains at least  $k_2$  drivers, where  $k_2$  is smaller than  $k_1$ . These new groups are established in such a way that the number of matched PPUF gates in each new partition is maximized. If the user with the correct response is legitimate, he can still easily calculate the response to verifier's challenge for the new matching. However, the attacker's probability is additionally reduced. Further, this procedure can be iteratively repeated until the probability of a successful attack is satisfactory. The key idea is that the number of matched gates in each group is maximally increased which allows for the reduction in anonymity to be favorably traded against the chances of a successful attack.

### D. Trusted Sensing and Computing

The quantitative advantages of PPUF-based cryptographic protocols in terms of metrics, such as time of execution, footprint, and energy, are important. They are

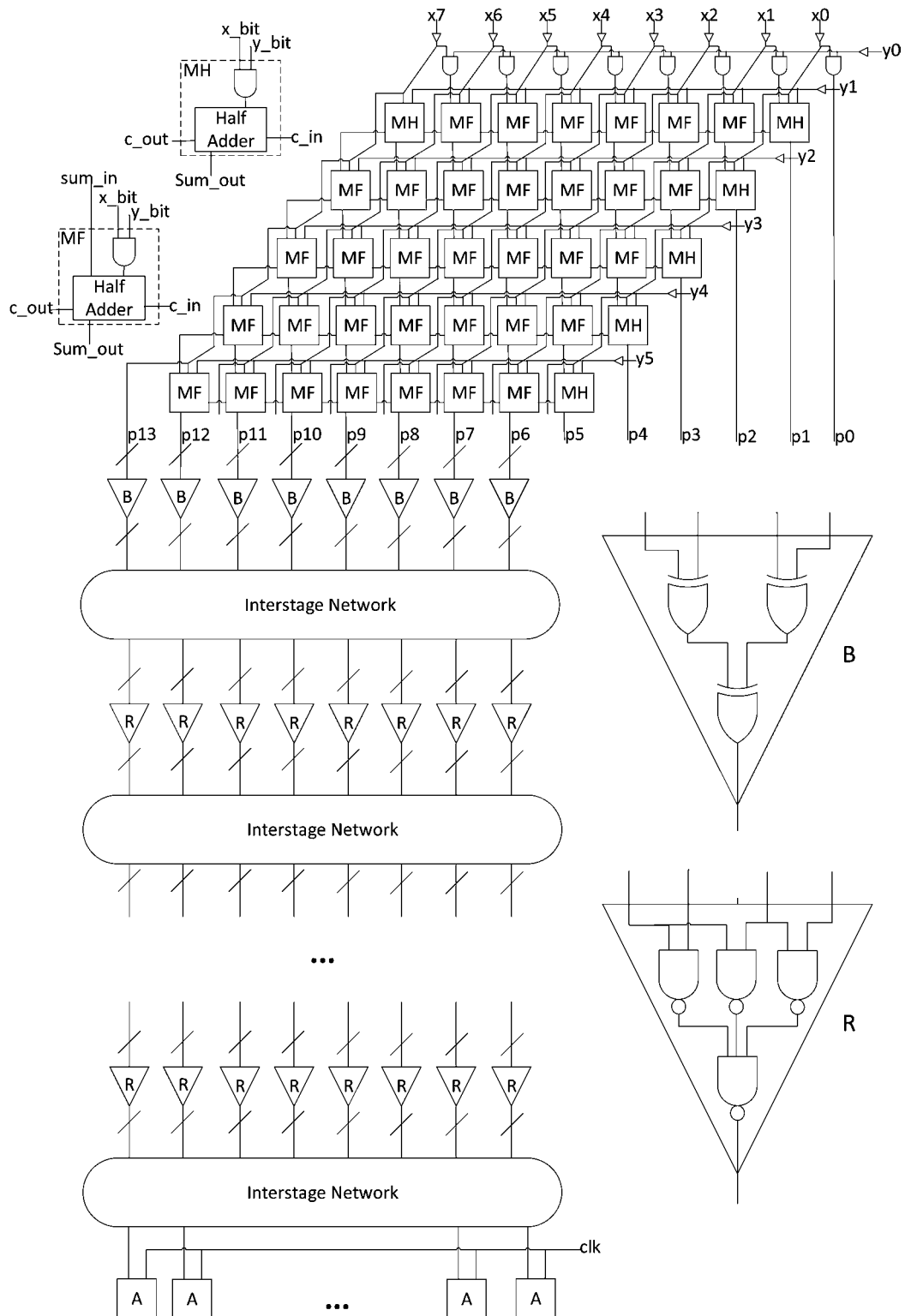


**Fig. 11.** Quantized matched PPUF integrated with multiplier circuit.

often three or more orders of magnitude more efficient in terms of energy. These advantages are further exaggerated by their resiliency against physical and side-channel attacks. However, these protocols will be serious alternatives to classical cryptography only if and when the security of these protocols is adequately tested, i.e., when numerous researchers launch numerous attacks.

From an application point of view and in a qualitative sense, the benefits of PPUFs are more apparent and practical. These benefits are particularly pronounced for quantized matched PPUFs. Specifically, there are two major advantages of matched PUFs using standardized quantization over other PPUF schemes. One is that it is the first security primitive (building block) that enables execution of public-key protocols in a single clock cycle while providing resiliency against physical and side-channel attacks. The other is that it enables integration with standard logic in such a way that the secure flow of information is guaranteed. This property can be used for the creation of a spectrum of new security, privacy, and trust systems, protocols, and applications. In the rest of this section, we focus on techniques that enable the creation of localized and distributed secure systems that employ quantized matched PPUFs. It is important to emphasize that these techniques can be applied to other types of PPUFs. However, these systems are significantly less efficient, except in the case of digital PPUFs.

Fig. 11 shows one way how we can integrate a standardized quantized PPUF with a multiplier. A part of the multiplier is used as part of the aged PPUF. The integration is realized in such a way that a subset of gates of the multiplier is simultaneously used for the computation of multiplications, and as part of the security primitive PPUF. As in Fig. 11, this may be achieved by treating the output of this subset as input to a quantized matched PPUF—multiple stages of booster (B) and repressor (R) cells that race against the clock to the arbiters (A). Therefore, if such a multiplier is used for computation, it automatically also participates in producing the PPUF output. The PPUF output, therefore, depends on the computation that is conducted by the multiplier and provides proof that a particular computation was indeed performed by the multiplier. Of course, an arbitrary arithmetic or logic unit may be used instead of the multiplier.

Similarly, instead of the computational or logic modules of processor, memory, or application-specific ICs, we can overlap PPUF circuitry with the circuitry of a sensor or a radio receiver to fulfill trusted sensing [24], [25] and communication requirements. Yet another option is to integrate an aged matched PPUF with clock circuitry. The key observation is that the attacker cannot alter any part of the system that is integrated with a matched PPUF without also altering the PPUF's properties. This is so because the delay characteristics of the pertinent PPUF are impacted by the driving loads of the integration gates.

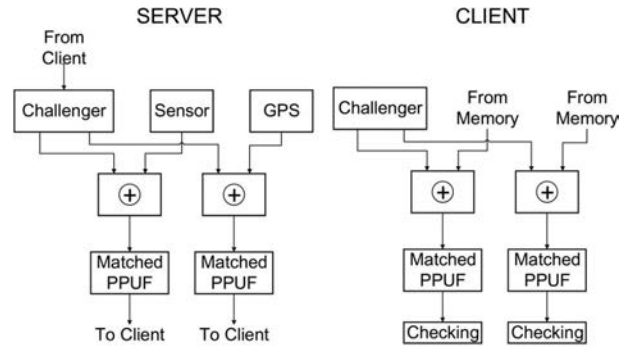


Fig. 12. Trusted sensing system and flow.

Using integrated matched PPUFs, one can create a variety of security, privacy, and trust protocols. We now discuss two such protocols. The first is for remote trusted sensing, where the goal is to design a distributed system that consists of two devices where the first receives trusted data from the second. The data are trusted in the sense that the first device can check that the received data are from the second device, and that it is collected at a specific time and a specific location. This is accomplished in the following way.

The distributed hardware platform is shown in Fig. 12. Both devices client (C) and server (S) have identical core architectures. However, while device S receives inputs to its matched PPUF from the GPS or sensors, device C is designed in such a way that its PPUF inputs can be independently controlled from regular memory. The GPS signal contains data that indicate the location and time of sampling (data collection) at device S. The server sends both original data collected from sensors and GPS as well as their corresponding matched PPUF output. The client can easily check the correctness of the data by comparing it with the values of its PPUF output for the claimed actual data. The protocol can be further improved to prevent replay attacks, if the data that are served to the inputs of the matched PPUFs are combined using XORing with a set of challenges that are supplied by the client.

In addition to the trusted remote sensing, the proposed scheme can also be utilized for remote trusted computation. Again, we have two participating parties: client (C) and server (S). It is assumed that the client has more restricted energy and computational resources, while the server provides services to the clients using its plentiful resources. For example, side C may be a smartphone and side S may be a data center. The client can check that the results or intermediate values calculated by the server are indeed calculated by the server at claimed times using a variant of the protocol for remote sensing. Hence, the client can probabilistically check the results and the required time by the data center. This checking can be done either in real time or offline, depending on the application and resources available at the client.

#### IV. TESTING PPUFs

In this section, we briefly discuss aspects related to testing PPUFs. We place equal emphasis on establishing relevant testing issues, the state-of-the-art testing techniques, and current performance of various types of PUFs and PPUFs.

Testing is an essential step in qualifying ICs for the actual usage. It has two main aspects. The first is manufacturing, where the goal is to check if the IC implementation was completed without faults. The second is functional testing, where the correctness of design is evaluated. Similarly, it is crucial that the design and implementation of PPUFs are properly evaluated before their deployment.

It is easy to see that IC manufacturing testing corresponds to gate and transistor level characterization (GLC) where delay, leakage, or some other device metrics are analyzed. GLC is a well-studied topic [26]. In order to address functional testing, one has to first define security attacks that are relevant to PPUFs. Currently, it appears that prediction and emulation attacks are the most effective. In prediction attacks, the attacker uses known challenge–response pairs to guess and accurately calculate the response for challenges of interest. In emulation attacks, the attacker tries to create or identify identical, or at least similar, PPUFs to facilitate subsequent prediction attacks.

Three specific types of prediction attacks have been undertaken with remarkable success on PUFs. The first is the use of linear algebra techniques, and in particular linear programming, to quickly create a PUF model [27]. The second type employs machine learning methods to build statistical models that accurately predict the response for random challenges with high probability. Finally, recently, it has been shown that side-channel-based techniques are exceptionally effective in reverse-engineering PUFs.

There are also a number of statistical metrics for predicting the expected level of effectiveness of any prediction attack. For example, it is highly desirable that correlation between any input and any output, or between any two outputs, be close to 0.5. It is also important that the dependency of any output on a small set of inputs and a small set of outputs is such that effective prediction is not possible. These statements should stay unchanged even if a set of intermediate signals are included in the set in order to avoid prediction attacks involving the calculation of intermediate signals and subsequently use them for the prediction of one or more outputs.

It is often exceptionally important that prediction is prevented for similar inputs, in particular for two inputs that have bitwise Hamming distance 1. This phenomenon is named the avalanche effect and is often studied in classical cryptography. It is well documented that early types of PUFs and PPUFs often had very weak avalanche effect properties. While ideally the Hamming distance of

the outputs should follow a uniform distribution, it often follows Gaussian distributions with a small variance. However, more recent PPUFs have significantly improved their avalanche effect distributions. One potential explanation is that some classes of PUFs and PPUFs are not sufficiently nonlinear and do not have a high enough share of mixing signals, which are two properties identified by Shannon as essential for cryptographic systems.

PPUFs can also be analyzed from a hardware point of view in terms of its predictability. For instance, one can study the extent to which two PPUFs are similar in terms of having a large percentage of identical corresponding gates that have identical responses to the same set of challenges. It would be natural to label the extreme situation where two PUFs do not match only on a single gate, a hardware avalanche effect. Preliminary simulation experiments show that it is much more difficult to find high-quality PPUF structures with respect to the hardware avalanche effect. At the same time, they show that more nonlinear PPUFs perform significantly better. We expect that the study of other hardware-based analyses of prediction and/or emulation capabilities will emerge as important criteria in the design of PUF and PPUF structures.

It is important to note that PPUF selection greatly depends on process variation and its assumed model. This dependency is complex. Too little variability makes measurements difficult. On the other hand, too much variability may also be detrimental. For instance, if one gate has a much higher delay than any other gate, it makes the PPUF an easy target for prediction attacks.

One important PPUF application is its use as a synchronized random number generator (RNG), i.e., the creation of two or more hardware RNGs with identical properties, so that no additional such RNG can be created. These synchronized RNGs can be used for a plethora of security and trust applications. Recently, Xu and Potkonjak [28] have demonstrated that such devices can be easily built using digital PPUFs. For example, the first digital PPUF design easily passed the complete NIST randomness test.

#### V. OPEN PROBLEMS

We finish our PPUF coverage by stating and explaining the rationale behind the most important pending challenges and opportunities. These challenges are separated to two research and development lines: PUF-related challenges and application-protocol-related problems.

##### A. PUF-Related Problems

1) *Stability*: For PPUF-based security primitives to see widespread use, their behavior must be stable under a wide range of operating and environmental conditions, which will endow the security primitive with the property of reliability and enhance their commercial viability. However, the first- and second-generation PPUF designs

can be highly sensitive to power supply noise and drops, thermal variations, and use-related device aging. These parameters can affect gate delays in ways that are complex and difficult to model. For example, this can quickly introduce inconsistencies in the output of matched PPUFs. Similarly, unintentional glitching may trigger arbiters before signals from the inputs actually arrive at the arbiters. It is, therefore, important to introduce techniques that can maintain the stability of the PPUF behavior despite variable operating and environmental conditions, in a manner that does not compromise fundamental PPUF properties such as the ESG, rapid execution times, low computational overhead, etc. Digital PPUFs are a step in this direction.

2) *Secrecy and Predictability*: A strong security characteristic of any security primitive is the predictability of its output given some input. The obvious approach here is to expose this characteristic via a set of security tests that prove unpredictability in the behavior of the security primitive. We can identify at least two such tests. 1) The distribution of the output bits should be scrutinized for patterns. For example, the entropy in PPUF output can be tested via compression techniques with theoretical guarantees such as the context-tree-weighting method [29]. The National Institute of Standards and Technology (NIST) battery of statistical tests may also be applied to check for nonrandomness. 2) The correlation between PPUF input and output should be examined. For example, the strict avalanche criterion tests this condition by checking if, when a single input bit is flipped, the output bits flip with probability 0.5. Such tests will be crucial in determining whether an attacker can gain an advantage merely by statistically modeling the input/output relationship in a computationally tractable manner.

## B. Application-Protocol-Related Problems

1) *Aging-Based Protocols*: Oblivious transfer protocols underlie an important class of cryptographic applications wherein a sender ( $S$ ) would like to send a message to a receiver ( $R$ ) such that  $R$  receives the message with probability 0.5, but  $S$  cannot ascertain whether  $R$  has received the message. For example, such a protocol has been shown to lend itself to commital-protocol-based secure multiparty communication schemes and zero knowledge proofs [30]. A simple-matching-based PPUF protocol may be devised to serve as an oblivious transfer protocol as follows:  $S$  communicates two sets of gates  $G_1$

and  $G_2$  that it may age.  $S$  and  $R$  independently pick one of the two sets and age the corresponding set of gates on their PUFs. The PUFs are now matched with probability 0.5.  $S$  generates a key based on the output of its PPUF which it uses to encrypt the message, and transfers it to  $R$ . Clearly,  $R$  can decrypt the message with probability 0.5. However, the problem with such aging-based security protocol designs is that device aging is a unidirectional operation and, after a few iterations, a gate may no longer be aged, effectively destroying the PPUF, or limiting its effectiveness, within just a few iterations. A solution to this problem has the potential of vastly expanding the range of protocols that PPUF-based primitives can support.

2) *Suceptibility to Side-Channel Attacks*: In Section III-D, we have discussed the application of PPUFs to trusted remote sensing and computing. However, in this context, PPUFs are extremely vulnerable making them susceptible to side-channel attacks. The attacker may have complete access to the PUF-integrated chip, including access to signals at the pins of the IC, revealing timing and power information among other things regarding the operations occurring on the IC. Simply encrypting all data before it is sent off chip (e.g., transfers to memory, disk, network, etc.) may not be sufficient for this purpose. Clearly, an important open problem in this context is the execution of computation and communication on PUF-integrated ICs while ensuring that no information regarding these secure operations is revealed via side-channels such as signal timing, system power usage, electromagnetic radiation, etc.

## VI. CONCLUSION

We have surveyed a recently emerging hardware security primitive: PPUFs. PPUFs support creation of secure communication, storage, sensing, and computing, as well as protocols for preserving privacy and ensuring trust. While currently the two largest PPUF families are gap-based and matching-based PPUFs, new families such as digital PPUFs have been proposed very recently. We have analyzed the properties of these PPUF families, discussed techniques for testing and evaluating PPUFs, and presented representative security protocols. The main quantitative advantages of PPUFs include energy efficiency, high throughput low latency, and a small footprint. The main qualitative advantages are their flexibility in the creation of new classes of security protocols, and their permanent integration with sensing and computing systems to enable trustable flow of information. ■

## REFERENCES

- [1] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, 1st ed. Boca Raton, FL: CRC Press, 1996.
- [2] N. Beckmann and M. Potkonjak, "Hardware-based public-key cryptography with public physically unclonable functions," in *Proc. Int. Workshop Inf. Hiding*, 2009, pp. 206–220.
- [3] U. Rührmair et al., "Towards electrical, integrated implementations of SIMPL systems," in *Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices*, ser. Lecture Notes in Computer Science, vol. 6033, Berlin, Germany: Springer-Verlag, 2010, pp. 277–292.
- [4] U. Rührmair, "SIMPL systems: On a public key variant of physical unclonable functions," 2009. [Online]. Available: <http://eprint.iacr.org/2009/255.pdf>
- [5] U. Rührmair and M. van Dijk, "On the practical use of physical unclonable functions

- in oblivious transfer and bit commitment protocols," *J. Cryptogr. Eng.*, vol. 3, no. 1, pp. 17–28, 2013.
- [6] U. Rührmair, "SIMPL systems as a keyless cryptographic and security primitive," *Cryptography and Security*. Berlin, Germany: Springer-Verlag, 2012, pp. 329–354.
- [7] R. Horstmeyer, B. Judkewitz, I. Vellekoop, and C. Yang, "Physical key-protected one-time pad," 2013. [Online]. Available: <http://arxiv.org/abs/1305.3886>
- [8] M. Potkonjak, S. Meguerdichian, A. Nahapetian, and S. Wei, "Differential public physically unclonable functions: Architecture and applications," in *Proc. Design Autom. Conf.*, 2011, pp. 242–247.
- [9] S. Meguerdichian and M. Potkonjak, "Matched public PUF: Ultra low energy security platform," in *Proc. Int. Symp. Low Power Electron. Design*, 2011, pp. 45–50.
- [10] M. Alam, H. Kufluoglu, D. Varghese, and S. Mahapatra, "A comprehensive model for PMOS NBTI degradation: Recent progress," *Microelectron. Reliab.*, vol. 47, no. 6, pp. 853–862, 2007.
- [11] S. Meguerdichian and M. Potkonjak, "Using standardized quantization for multi-party PUF matching: Foundations and applications," in *Proc. Int. Conf. Comput.-Aided Design*, 2012, pp. 577–584.
- [12] B. Cline, K. Chopra, D. Blaauw, and Y. Cao, "Analysis and modeling of CD variation for statistical static timing," in *Proc. Int. Conf. Comput.-Aided Design*, 2006, pp. 60–66.
- [13] D. Markovic, C. Wang, L. Alarcon, T.-T. Liu, and J. Rabaey, "Ultra low-power design in near-threshold region," *Proc. IEEE*, vol. 98, no. 2, pp. 237–252, Feb. 2010.
- [14] A. Asenov, "Random dopant induced threshold voltage lowering and fluctuations in sub-0.1  $\mu\text{m}$  MOSFETs: A 3-D atomistic simulation study," *IEEE Trans. Electron Devices*, vol. 45, no. 12, pp. 2505–2513, Dec. 1998.
- [15] S. Chakravarthi, A. Krishnan, V. Reddy, C. Machala, and S. Krishnan, "A comprehensive framework for predictive modeling of negative bias temperature instability," in *Proc. IEEE Int. Reliab. Phys. Symp.*, 2004, pp. 273–282.
- [16] T. Xu, J. Wendt, and M. Potkonjak, "Digital bimodal function: An ultra-low energy security primitive," in *Proc. Int. Symp. Low Power Electron. Design*, 2013, pp. 292–297.
- [17] T. Xu and M. Potkonjak, "Lightweight digital hardware random number generators," in *Proc. IEEE SENSORS*, 2013, DOI: 10.1109/ICSENS.2013.6688562.
- [18] T. Xu, W. J. B., and M. Potkonjak, "Matched digital PUFs for low power security in implantable medical devices," in *Proc. Int. Conf. Healthcare Inf.*, Verona, Italy, 2014, to be published.
- [19] T. Xu and M. Potkonjak, "Robust and flexible FPGA-based digital PUF," in *Proc. Int. Conf. Field Programmable Logic Appl.*, Munich, Germany, 2014, to be published.
- [20] D. A. Patterson, "Latency lags bandwidth," *Commun. ACM*, vol. 47, no. 10, pp. 71–75, 2004.
- [21] J. Wendt and M. Potkonjak, "The bidirectional polyomino partitioned PUF as a hardware security primitive," in *Proc. Global Conf. Signal Inf. Process.*, 2013, pp. 257–260.
- [22] Z. Zhu and G. Cao, "Applaus: A privacy-preserving location proof updating system for location-based services," in *Proc. IEEE INFOCOM*, 2011, pp. 1889–1897.
- [23] D. Liu, P. Ning, and W. K. Du, "Attack-resistant location estimation in sensor networks," in *Proc. Int. Symp. Inf. Process. Sensor Netw.*, 2005, pp. 99–106.
- [24] M. Potkonjak, S. Meguerdichian, and J. Wong, "Trusted sensors and remote sensing," in *Proc. IEEE SENSORS*, Nov. 2010, pp. 1104–1107.
- [25] J. Wendt and M. Potkonjak, "Nanotechnology-based trusted remote sensing," in *Proc. IEEE SENSORS*, 2011, pp. 1213–1216.
- [26] S. Wei, S. Meguerdichian, and M. Potkonjak, "Gate-level characterization: Foundations and hardware security applications," in *Proc. Design Autom. Conf.*, 2010, pp. 222–227.
- [27] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *Proc. IEEE Int. Test Conf.*, 2008, DOI: 10.1109/TEST.2008.4700636.
- [28] T. Xu and M. Potkonjak, "Lightweight digital hardware random number generators," in *Proc. IEEE SENSORS*, 2013, DOI: 10.1109/ICSENS.2013.6688562.
- [29] T. Ignatenko, G.-J. Schrijen, B. Skoric, P. Tuyls, and F. Willems, "Estimating the secrecy-rate of physical unclonable functions with the context-tree weighting method," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2006, pp. 499–503.
- [30] J. Kilian, "Founding cryptography on oblivious transfer," in *Proc. 20th Annu. ACM Symp. Theory Comput.*, 1988, pp. 20–31.

## ABOUT THE AUTHORS

**Miodrag Potkonjak** (Member, IEEE) received the Ph.D. degree in electrical engineering and computer science from the University of California Berkeley, Berkeley, CA, USA, in 1991.

He is a Professor with Computer Science Department, University of California Los Angeles (UCLA), Los Angeles, CA, USA. He created first watermarking, fingerprinting, and metering techniques for integrated circuits, as well as first remote trusted sensing and trusted synthesis approaches, compilation using untrusted tools, and public physical unclonable functions.

**Vishwa Goudar** (Student Member, IEEE) received the Ph.D. degree in computer science from the University of California Los Angeles (UCLA), Los Angeles, CA, USA, in 2013.

His research interests lie in the area of security, fault tolerance, and power management of low-power wearable systems.

# Microcontrollers as (In)Security Devices for Pervasive Computing Applications

*This paper discusses possible threats to embedded systems using two microcontroller case studies.*

By DAEHYUN STROBEL, DAVID OSWALD, BASTIAN RICHTER, FALK SCHELLENBERG, AND CHRISTOF PAAR, *Fellow IEEE*

**ABSTRACT** | Often overlooked, microcontrollers are the central component in embedded systems which drive the evolution toward the Internet of Things (IoT). They are small, easy to handle, low cost, and with myriads of pervasive applications. An increasing number of microcontroller-equipped systems are security and safety critical. In this tutorial, we take a critical look at the security aspects of today's microcontrollers. We demonstrate why the implementation of sensitive applications on a standard microcontroller can lead to severe security problems. To this end, we summarize various threats to microcontroller-based systems, including side-channel analysis and different methods for extracting embedded code. In two case studies, we demonstrate the relevance of these techniques in real-world applications: Both analyzed systems, a widely used digital locking system and the YubiKey 2 onetime password generator, turned out to be susceptible to attacks against the actual implementations, allowing an adversary to extract the cryptographic keys which, in turn, leads to a total collapse of the system security.

**KEYWORDS** | Code extraction; microcontroller; real-world attacks; reverse engineering; side-channel analysis

## I. INTRODUCTION

A surprisingly large share of the world's central processing unit (CPU) market still goes to low-end, off-the-shelf 8-b  $\mu$ Cs. They combine several versatile modules in a single chip, e.g., a processor unit, program and data memory, clock generation and peripherals, and are therefore often called a self-contained computing system. Corresponding embedded applications range from household appliances over car electronics to industrial control. A steadily increasing number of  $\mu$ C applications require security. Examples include contactless and traditional smart cards, electronic control units (ECUs) in cars, tokens for access control, industrial machine controllers (for services such as software download or feature activation), and even medical implants. The continuing trend toward the Internet of Things (IoT) will only increase the need for embedded applications with strong security features. It is thus of great interest for design engineers and users to assess the feasibility of  $\mu$ Cs as security devices.

In this tutorial, we will discuss techniques to attack (cryptographic) implementations running on  $\mu$ Cs. First, we introduce side-channel analysis (SCA) in Section II. SCA allows breaking mathematically secure ciphers by analyzing physical leakages observed during the execution on a field-programmable gate array (FPGA), an application-specific integrated circuit (ASIC), or a  $\mu$ C. SCA has its origin in the late 1990s, when Kocher, Jaffe, and Jun introduced the first attacks based on the timing behavior and power consumption of cryptographic implementations [1], [2]. It turns out that 8-b  $\mu$ Cs are particularly susceptible against SCA. For example, extracting the key of an unprotected Advanced Encryption Standard (AES) implementation (AES is the world's most widely used

Manuscript received September 2, 2013; accepted May 1, 2014. Date of publication June 5, 2014; date of current version July 18, 2014. This work was supported in part by the German Federal Ministry of Education and Research BMBF under Grant 01IS10026A, Project EXSET.

The authors are with the Horst Görtz Institute for IT-Security, Ruhr University Bochum, Bochum 44801, Germany (e-mail: daehyun.strobel@rub.de; david.oswald@rub.de; bastian.richter@rub.de; falk.schellenberg@rub.de; christof.paar@rub.de).

Digital Object Identifier: 10.1109/JPROC.2014.2325397

0018-9219 © 2014 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See [http://www.ieee.org/publications\\_standards/publications/rights/index.html](http://www.ieee.org/publications_standards/publications/rights/index.html) for more information.

encryption algorithm) on an Atmel AVR ATmega8 is feasible with less than 50 power measurements under ideal conditions. Masking and hiding methods are possible countermeasures to complicate side-channel attacks, but often cannot prevent them completely [3].

The next topic of this tutorial is concerned with extracting and reverse engineering of embedded program code of a  $\mu\text{C}$  (cf. Section III), which can be a serious threat for some applications. Nearly all of today's  $\mu\text{C}$ s have some mechanism for code protection. In most cases, this is a lock bit that, if enabled, shall ensure that the firmware and data stored on the device cannot be read by an unauthorized party. This feature is used by numerous manufacturers to protect intellectual property and to assure safety and security. Of course, this assumes that the protection cannot be circumvented. However, research has shown that this assumption is not valid for several devices. For instance, Skorobogatov investigated several  $\mu\text{C}$ s with respect to their susceptibility against invasive, semi-invasive, and noninvasive attacks. The result was that the code protection of several  $\mu\text{C}$ s from different manufacturers could be bypassed with power glitching techniques, microprobing, exposure to UV-C light, or by fault injection [4]. Although this report was published in 2005, reliably protecting embedded code still appears to be an open problem, as one of our case studies shows.

In Section IV, we describe an entirely different approach to bypass the readout protection. It is possible to determine the executed instruction sequence of a CPU on the basis of the side-channel leakage. The adversary's goal is to map side-channel waveforms to the corresponding  $\mu\text{C}$  instructions. This can be achieved with linear [5] or nonlinear [6] pattern recognition algorithms, using principal component analysis (PCA) and Fisher's linear discriminant analysis (LDA) dimensionality reduction on the one hand, and support vector machines (SVMs) on the other hand. Both linear and nonlinear approaches lead to a recognition rate of approximately 70% on a PIC16F687, i.e., on average, 70% of the instructions of a program were recovered successfully. The recognition rate can be further improved using Markov chains or, for individual instructions, using boosting techniques for SVMs.

In Sections V and VI, we present two penetration tests against real-world applications, both equipped with standard  $\mu\text{C}$ s. The attack of Section V targets the digital locking and access control system SimonsVoss 3060 G2 [7]. SimonsVoss is the European market leader for electronic access control systems [8] and has sold more than 1 000 000 digital locking cylinders and 2 000 000 corresponding transponders [9]. The system 3060 is the electronic equivalent of a mechanical locking system. The term G2 identifies the latest hardware generation, which relies on a previously undisclosed, proprietary cipher and authentication protocol between transponder and digital cylinder. On both components, a Microchip PIC16F886  $\mu\text{C}$  is used to implement the authentication process. Al-

though the access to the program code for the protocol and the cryptographic algorithms was locked, the readout protection could be disabled by decapsulating the chip and exposing parts of it to UV-C light. After that, the complete program code and data of the door lock and the transponder were extracted and analyzed. Based on the reverse-engineered protocol, both cryptanalytical and side-channel attacks were developed that allow an adversary to bypass the locking system and gain unauthorized access. Section V is based on the research presented in [7] and [10].

The second security analysis (Section VI) targets Yubico's YubiKey 2 tokens [11]. They are used for a two-factor authentication, i.e., access is granted only if the user 1) knows the correct password and 2) is in physical possession of the corresponding hardware token. Typical applications are password management or computer login. Even though not a direct IoT application, it demonstrates how weaknesses of an embedded  $\mu\text{C}$ s can be a threat for computer network security. The YubiKey 2 provides an AES-encrypted onetime password (OTP) in addition to the standard username/password credentials. This OTP is derived on an 8-b  $\mu\text{C}$  manufactured by Sunplus Innovation Technology Inc. (Taiwan). As described in [12], by noninvasively measuring the electromagnetic (EM) emanation of the device, it is possible to extract the full 128-b AES key by means of SCA in less than 1 h. The attack leaves no physical traces on the device and can be performed using low-cost equipment. In consequence, an adversary is able to generate valid OTPs, even after the YubiKey 2 has been returned to the owner.

## II. SIDE-CHANNEL ANALYSIS

SCA has gained considerable attention since its introduction in 1996 [1]. One reason for this is that many real-world targets were broken due to their susceptibility to side-channel attacks (e.g., [10] and [12]–[15]).

Typically, a SCA is performed in two steps. First, the adversary has physical access to the target device and acquires a side-channel signal (e.g., the power consumption or the EM emanation) during the cryptographic computation. The resulting signal is often called side-channel trace or simply trace. In the second phase, the acquired traces are evaluated using digital signal processing techniques and statistical methods in order to recover a (cryptographic) secret. In Section II-A–II-C, we summarize the most common evaluation methods proposed in the literature.

### A. Simple Power Analysis

In a simple power analysis (SPA) [2], one trace or an average over several traces (with identical input) to reduce the noise is usually visually inspected to directly derive a cryptographic secret. For example, let us consider a square-and-multiply algorithm to compute a Rivest–Shamir–Adleman (RSA) signature. This exponentiation algorithm consists of square operations and, depending on

the secret key, multiple multiplication operations. Generally, a squaring can be implemented more efficiently than a multiplication. This results in, e.g., a lower cycle count or a lower energy consumption. If an adversary can distinguish a multiplication from a squaring in a trace, the secret exponent could directly be read off. SPA can be applied to a variety of cryptographic algorithms, including the AES key schedule [16] and elliptic curve cryptography (ECC) scalar multiplication [17]. Additionally, SPA can be used when profiling a device under test (DUT) with an unknown implementation to prepare a key recovery. For instance, cryptographic operations often lead to an increased power consumption and can thus be identified in a trace, reducing the amount of points in time to be evaluated.

## B. Template Attacks

Template attacks (TAs) are essentially an extension of SPA, replacing the manual inspection with pattern matching and machine learning techniques. In contrast to SPA, a TA requires a profiling phase, i.e., a step during which the DUT is under full control of the adversary to estimate the statistical relation between the observable random variables—in our case the respective points in time of a side-channel trace—and the internal states that are to be distinguished (for example, the value of a key byte). The resulting training set is then used to recover the desired values from a test set, i.e., traces for which the value of the key byte is considered unknown. Further details on TAs (in the context of SCA for code extraction) are given in Section IV.

## C. Correlation Power Analysis

In contrast to SPA, differential power analysis (DPA) [2] and correlation power analysis (CPA) [18] are based on the evaluation of many traces with varying input data for the targeted algorithm. A brute-force attack with additional information is performed on a part of the algorithm. For instance, a DPA attack on the Data Encryption Standard (DES) or AES usually only targets the first round, for which each S-box is processed separately. Hence, each subkey entering one specific S-box can be recovered independently from the remaining key bits. By testing all possible candidates for a given subkey (e.g.,  $2^6 = 64$  for the DES or  $2^8 = 256$  for the AES), the full key can be extracted in a divide-and-conquer manner.

For the attacks, an adversary first acquires  $N$  traces  $x_i$ ,  $i \in \{0, \dots, N-1\}$ , with varying input data  $M_i$ . To recover the cryptographic key, the traces are then statistically processed, e.g., using the Pearson correlation coefficient when performing a CPA.

Let  $K$  be the full key space of a block cipher. The adversary fixes a (small) subset  $K_{\text{cand}} \subseteq K$  (e.g., the 256 possible 8-b subkeys entering one S-box of the AES) and considers all key candidates  $k \in K_{\text{cand}}$ . Then, for each  $k \in K_{\text{cand}}$  and for each  $i$ , a hypothesis  $V_{k,i}$  on the value of

some intermediate (e.g., the output of one 8-b AES S-box) is computed. Using a power model  $f$ , this value is then mapped to  $h_{k,i} = f(V_{k,i})$  to describe the process that causes the side-channel leakage. In practice, a Hamming weight (HW) or a Hamming distance (HD) power model is often suitable for complementary metal–oxide–semiconductor (CMOS) devices such as  $\mu\text{Cs}$  [3].

In order to detect the dependency between  $h_{k,i}$  and  $x_i(t)$ , the correlation coefficient  $\rho_k(t)$  (for each point in time  $t$  and each key candidate  $k \in K_{\text{cand}}$ ) is given as

$$\rho_k(t) = \frac{\text{cov}(x(t), h_k)}{\sqrt{\text{var}(x(t))\text{var}(h_k)}}$$

with  $\text{var}(\cdot)$  indicating the sample variance and  $\text{cov}(\cdot, \cdot)$  the sample covariance according to the standard definitions [19]. The key candidate  $\hat{k}$  with the maximum correlation  $\hat{k} = \arg \max_{k,t} \rho_k(t)$  is assumed to be the correct secret key. Considering, for example, an implementation of the AES, the attack can be performed for each S-box and the corresponding subkey separately. Instead of a complexity of  $\mathcal{O}(2^{128})$  to recover the full 128-b key by an exhaustive search, this side-channel attack lowers the complexity down to  $\mathcal{O}(16 \cdot 2^8)$ .

## D. Side-Channel Leakage of $\mu\text{Cs}$ compared to FPGAs and ASICs

The fundamental prerequisite for side-channel attacks is the assumption that the DUT consumes power depending on the currently processed value. For a power analysis, a basic measuring setup uses a shunt resistor in the supply path to measure the current drawn by the DUT. As an alternative, the EM emanation can be acquired by placing an EM probe close to the DUT. The physical reason for the data-dependent current consumption or EM emanation is the switching of transistors and their loads. Assuming a CMOS design, the total consumption is the sum of the static leakage current and the (for SCA more relevant) dynamic switching current. The dynamic switching current consists of the short-circuit current and the current required to charge the load capacitance of a node when the output changes from low to high. Most prominent, however, is the current caused by the activity of all attached cells, i.e., the whole combinatorial circuit switching because of a single changed bit at one input.

Usually, a  $\mu\text{C}$  includes a central bus connecting different distinct elements on the silicon die, e.g., to connect the computational unit to the registers, nonvolatile memory, and other peripherals. In every clock cycle, the processed intermediate value is written on the bus to be fed into the arithmetic logic unit (ALU), etc., causing current consumption to charge the bus and the load. In contrast, as ASICs or FPGAs implement the functionality directly in hardware, there is usually no requirement for a general

purpose arithmetic unit and consequent registers that have to be addressed through a bus by the memory.

In [20], different side-channel evaluation methods for the SASEBO GII (cf., [21]) built around a Xilinx Virtex-5 FPGA are compared in the form of a contest. The best attack within the contest required around 7000 traces, and later, a TA (cf. Section II-B) was presented that succeeded with only 439 traces (cf., [22]).

In contrast, for a straightforward CPA of an AES implementation on an Atmel ATmega32  $\mu$ C, the full 128-b key can be recovered with approximately 50 traces—an order of magnitude lower compared to the highly optimized attack on the FPGA implementation. In general,  $\mu$ Cs inherently exhibit a stronger side-channel leakage compared to logic implemented in hardware.

### III. CODE EXTRACTION

Embedded devices are inherently exposed to product counterfeit. The main problem with protecting against this treat is that an attacker has unlimited physical access to the device and can deeply investigate its design and functionality in order to copy it. This has special relevance for devices fulfilling a security purpose like encryption or authentication: Understanding or getting to know the internal details is usually mandatory for mathematical analysis or side-channel attacks (cf., Section II).

Whenever  $\mu$ Cs are used as central computational unit, most of the intellectual property (IP) is contained within the program code. Therefore, nearly all manufactures of  $\mu$ C implement methods to protect the internal memories from being readable. Those methods include single nonvolatile storage cells blocking access, physically destroyable programming interfaces (cf., [23]), or even a password protection, allowing only authorized persons to read the code (e.g., [24]).

In 2005, Skorobogatov presented a comprehensive report [4] investigating ways to circumvent or disable readout protection features. As it turned out, the protection of multiple devices of various manufactures can be bypassed using attacks with varying complexity and invasiveness. For example, some devices are vulnerable to variations of the power supply or the clock input—not requiring direct access to the silicon die. As shown in [25], even the programing interface might be logically flawed. Semi-invasive attacks directly access the silicon die, enabling optical glitches like stimulation with lasers, UV-C light, or simple photo flashes. Certainly the most sophisticated and expensive attacks belong to the fully invasive class. Here, the attackers directly alter or probe individual buses or logic cells.

In the following, we focus on products of Microchip Technology Inc. (USA), which recently delivered its 12 billionth  $\mu$ C [26] and is ranked second place based on the marked share of 8-b  $\mu$ Cs according to [27]. Following the shipment figures at [28], 8-b  $\mu$ C can still be considered a

very pervasive architecture today. Microchip divides its (flash-based) 8-b devices into the series 10F, 12F, 16F, and 18F, representing different internal architectures including varying instruction sets.

In [25], Meriac exploits a flaw in the programming interface in the PIC18FXX2/XX8 series to obtain the program code of a commercial device. These chips feature a segmented program memory with a protection bit for each segment. However, Microchip allows overwriting single segments even if the code protection is set. Consequently, it is possible to overwrite a segment with code that is able to dumb the internal code in a serial manner, circumventing the readout protection of the remaining unmodified segments. A similar program interface can be found for other models of the PIC18F series.

A semi-invasive attack on a PIC18F1320 and a PIC12F683 involving UV-C light is described in [29] and [30]. The attack is based on the method to erase today's outdated erasable programmable read only memories (EPROMs) by shining UV-C light through a small window in the package. Here, the light stimulation will erase the charge of floating gate transistors. Microchip also produced such EPROMs and, thus, knew that the configuration bits disallowing to read the program code have to be protected against UV-C light. To this end, Microchip covered the floating gate transistors with small metal plates. However, as discussed in [29], the metal plates are too small, and when shining the UV-C light at a certain angle, it will bounce off various parts, eventually hitting the floating gate.

We used this idea to test whether other models of the 10F, 12F, and 16F series are vulnerable. For example, Fig. 1(a) depicts a decapsulated PIC16F687. Following [30], the area of configuration bits as highlighted in Fig. 1(b) is depicted in Fig. 1(c). Other directly identifiable areas include the memories on the left-hand side of Fig. 1(b): the flash memory in the center, EEPROM at the top right next to the configuration bits, and static random-access memory (SRAM) in the bottom-left corner. Note that, even without using a microscope, the locations of the memories and the configuration register are distinguishable from the remaining parts of the  $\mu$ C (cf., Fig. 1). Thus, after identifying the relevant areas, it is possible to cover the memories in order to protect them from UV-C exposure without any magnification [cf., Fig. 1(d)]. Finally, an exposure of the silicon die to UV-C light for about 20 min resets the whole configuration register, allowing dumping the memory.

As listed in Table 1, we found that nearly all of our randomly chosen test sets of the 10F, 12F, and 16F Microchip series are vulnerable to this kind of attack. Rather, solely outdated models like the 16F84(A) were found to be resistant to UV-C light. However, Skorobogatov [4, p. 61] indicates that those  $\mu$ Cs are vulnerable to a noninvasive voltage glitch during the erase operation. Although there is a large list of  $\mu$ C currently available from Microchip, the design is widely modular and only differs in size of

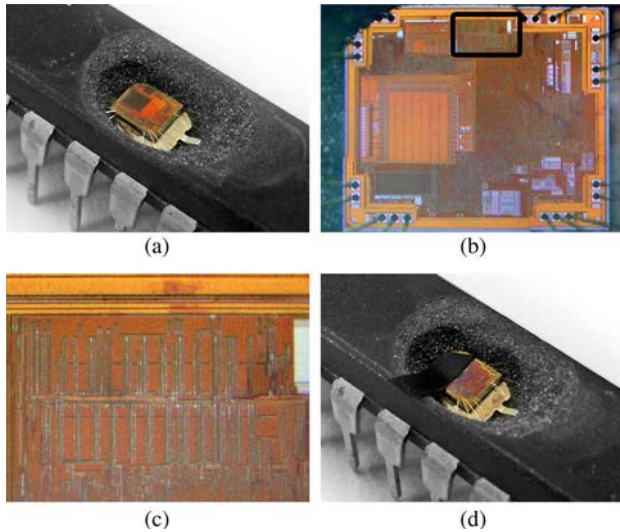


**Fig. 1.** Code extraction with UV-C light of a PIC16F687: (a)  $\mu$ C decapsulated using white fuming nitric acid (WFNA); (b) silicon die, configuration register highlighted; (c) detail of configuration register; and (d) nonvolatile memories protected against UV-C light by isolation tape.

memory or number of analog-to-digital converter (ADC) channels, etc. Thus, the idea of using exposure to UV-C light to reset the security mechanism is presumably applicable for most of the 10F to 16F series of Microchip. Using such a device possesses a large threat concerning product counterfeit.

#### IV. SIDE-CHANNEL CODE EXTRACTION

While most of the publications dealing with SCA focus on extracting the key of a cipher implementation, there exist also a few publications that concentrate on retrieving information about the executed code of a  $\mu$ C solely by observing its power consumption [5], [6], [31]. Interesting fields of applications are, e.g., to overcome the IP protection realized by the readout fuse bit of the  $\mu$ C or, from a legal point of view, the detection of product piracy. As a target device, all mentioned papers use the 8-b Microchip PIC16F687 with an instruction set of 35 instructions and an opcode size of 14 b. The core idea is to apply machine learning techniques to reconstruct the executed code, using the principles of TAs (cf., Section II-B). The code extraction problem is considered as a classical classification task which is achieved in mainly two phases.

**Table 1** Microchip  $\mu$ C of the 10F, 12F, and 16F Series Found to Be Vulnerable to an UV-C Attack

|        |         |          |             |        |
|--------|---------|----------|-------------|--------|
| 10F200 | 12F519  | 12F629   | 12F683 [30] | 16F73  |
| 16F88  | 16F627A | 16F628A  | 16F687      | 16F716 |
| 16F818 | 16F886  | 16LF1826 |             |        |

#### A. Phase I: Preprocessing

In this phase, a large number of traces are recorded for every instruction. Note that by using the term instruction, we only consider the instruction itself, e.g., ADDLW, MOVF, etc., with varying or randomized operands. This results in 41 sets of traces,<sup>1</sup> which is called the training set and forms the basis for the preprocessing phase. The role of preprocessing is to highlight important distinctive features that help us to distinguish between individual instructions and to discard background information or noise, which is not necessary for the classification process and would therefore only increase the computational workload. To give an example, most instructions on the PIC16F687 need four clock cycles for the execution. Depending on the instruction, within these four clock cycles, the value of the working register is read in, the computation is done, the result is stored, and the next instruction is prefetched from the instruction memory. As one can imagine, most of this information is insignificant regarding the main task of classifying the executed instruction. Thus, the goal is to focus only on relevant parts of the recordings that maximize the success with minimal effort. This can be realized by applying either linear or nonlinear feature extraction algorithms. Prominent linear feature extraction algorithms are, e.g., PCA and Fisher's LDA, and nonlinear algorithms include, e.g., kernel methods.

In a normal scenario, the data to be analyzed comprise a multitude of redundancies. In a trace, for example, the points around a peak are strongly correlated and add only little information to the amplitude of the peak. The technique of PCA tries to minimize these correlations by transforming the data into a subspace with a new basis. The axes are chosen in such a way that the first axis points to the direction of the largest variance, the second one to the largest variance under the constraint that the two axes are perpendicular, and so on. The reduction takes place by choosing only axes that are based on high variances. In [5], it is shown that considering more than 16 dimensions (compared to a few thousand dimensions before the reduction) does not improve the classification result significantly.

One disadvantage of the standard PCA is that it does not incorporate any information about the different classes. A workaround for this issue is proposed in [32] by applying a class-mean-PCA. Instead of computing the PCA of all data, it is only applied on the class means to maximize the variances between these classes. In addition to the variances between the classes, Fisher's LDA also tries to minimize the variances inside the classes. Fig. 2 shows a comparison between these three linear algorithms.

In contrast to [5], Malysiak [6] tries to solve the classification problem by using nonlinear algorithms. For

<sup>1</sup>Twenty five one-cycle instructions and eight two-cycle instructions, excluding the specific instructions SLEEP and RETFIE. For two-cycle instructions, every clock cycle is analyzed separately.

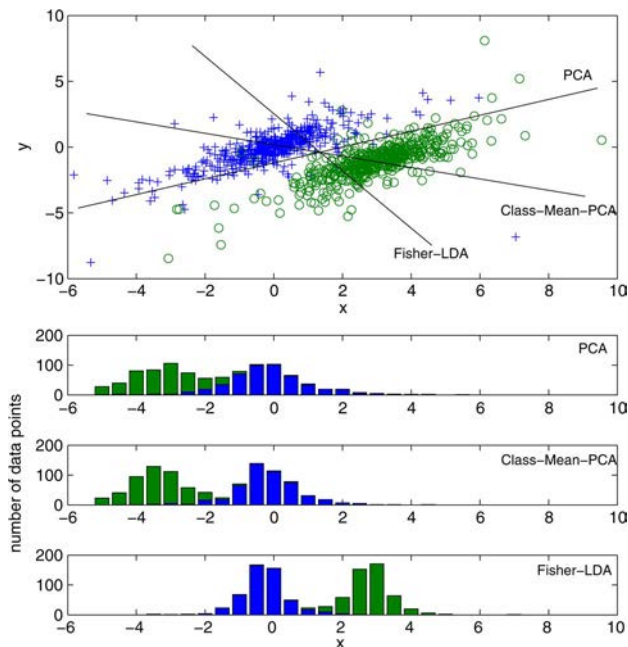


**Fig. 2. Comparison between PCA, class-mean-PCA, and Fisher's LDA.** The original data set has two axes  $x$  and  $y$ . The lines in the upper plot illustrate the resulting 1-D subspaces. The overlap of the data in the subspaces is shown in the histograms below. In this example, PCA leads to a high overlap of the different classes. Better results can be achieved by considering the variances between and inside the classes [5].

example, as a nonlinear feature extraction algorithm, the kernel PCA (KPCA) is applied instead of the linear PCA. In its core, KPCA works just like PCA. In cases where different classes cannot be separated linearly, the given data points are first transformed from the input space to a higher dimensional feature space before performing PCA computations. The idea behind this is that by choosing an appropriate transformation, the feature space allows again a linear separation of the data. This transformation is often referred to as the kernel trick.

## B. Phase II: Classification Task

The second phase is the actual classification task. In this phase, newly recorded traces are assigned to specific classes using an optimal feature subset and an appropriate classifier. A simple and intuitive approach of designing a classifier is based on finding similarities between the trace of the instruction to be categorized and the training data. In [5], this is done by building templates for each instruction class composed of a class mean and a covariance matrix. A new observation is then assigned to the class with the highest posterior probability. Combined with the linear feature extraction algorithms mentioned above, this approach yields to comparable results: While the standard PCA with a dimension of 16 leads to a recognition rate of 65.2%, the class-mean-PCA results in an average rate of



**Fig. 3. Components of the digital locking system. (a) Transponder 3064. (b) Door lock cylinder 3061.**

66.5% with 20 dimensions and Fisher's LDA in 70.1% using 17 dimensions.

In [6], an SVM approach achieves the best results, outperforming two other classification attempts, KPCA in combination with a kernel  $k$ -nearest neighbor classification and a kernelized version of Fisher's LDA. Roughly speaking, if the classes are not linearly separable, the SVM makes use of the kernel trick. After the transformation of the data to an adequate higher dimensional feature space, a hyperplane or a set of hyperplanes are constructed that linearly separate the data with maximum distance.

The average recognition rate of the chosen SVMs is comparable to the results of the linear Fisher's LDA (70% using the same data<sup>2</sup>). At first glance, SVMs therefore provide no significant benefit over linear techniques which are less computational expensive. However, the SVM approach leaves much space for optimizations. In [6], a boosting technique is proposed that combines several SVMs with different recognition rates of the individual classes. For instance, while the optimal SVM with an average recognition rate of 70% achieves poor results on classifying XORWF instructions, another SVM with a lower average rate might reach better results on this instruction. The boosting technique is hence a promising approach to increase the overall recognition rate by combining the results of several SVMs.

## V. CASE STUDY I: THE SIMONSVOSS DIGITAL LOCKING SYSTEM

SimonsVoss Technologies AG (Germany) is the European market leader for electronic locking and access control systems [8]. The list of customers and objects secured with this technology in Europe, the United States, and Asia, as listed on the official website [33], includes residential buildings, tourist apartments, hospitals, universities, embassies, major banks, airports, buildings of the German armed forces and the U.S. army, factory sites of well-known brands, police stations, stadiums, town halls, prisons, insurances, and many others.

<sup>2</sup>As denoted in [6], 64.1% refers to another data set of a different  $\mu C$ .

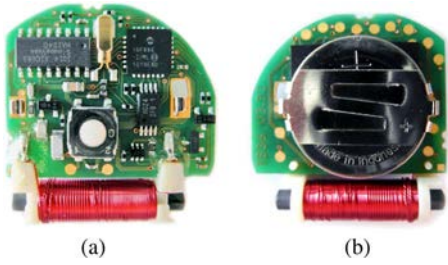


**Fig. 4.** PCB of transponder 3064. (a) Front. (b) Back.

The widespread digital locking system 3060 analyzed in [7] and [10] is based on an undisclosed, proprietary cryptographic protocol. The latest revision is termed “Generation 2” or “G2-based” system by the manufacturer. It supports up to 64 000 digital locking cylinders 3061 [cf., Fig. 3(b)] per installation, up to 64 000 transponders 3064 [cf., Fig. 3(a)] per lock, and the storage of up to 1000 access instances on the transponder.

The back-end of larger installations is realized as software running on a standard personal computer (PC), allowing configuring all door locks of an installation via a wireless link, e.g., in order to reprogram the door locks. Likewise, transponders can be programmed to enable access to certain doors by means of the functionality of the back-end. In general, the “Generation 2” system enables to form a network of door cylinders, transponders, and the back-end by means of various communication techniques, e.g., through Ethernet or through multiple routers and nodes (cf., [34]). Small installations can also be configured offline, using the programming transponder 3067.

When initially analyzing the SimonsVoss system 3060, we were facing a complete black box, i.e., we had no information on the inner workings of the system. From publicly available information, little can be learned about the actual implementation. Hence, we decided to obtain the necessary knowledge for our security analysis by reverse engineering the involved components. The printed circuit boards (PCBs) of transponder and door (cf., Figs. 4 and 5) have a similar layout containing three main components that are involved in the authentication process: A

SimonsVoss-proprietary ASIC is mainly responsible for modulating and demodulating the radio-frequency (RF) transmission. It is connected to a Microchip PIC16F886  $\mu\text{C}$  [35], which holds the implementation of the authentication protocol. The third component is an external electrically erasable programmable read-only memory (EEPROM) controlled by the  $\mu\text{C}$  over an inter-integrated circuit (I<sup>2</sup>C) bus [36].

#### A. Reading Out the Firmware of the PIC16F886

As described in Section III, the PIC stores its firmware, i.e., the embedded code that determines the behavior of the  $\mu\text{C}$ , in an internal flash memory. Moreover, the  $\mu\text{C}$  contains an internal EEPROM for storing 256 bytes (B) of user-defined data. In our case, after unsoldering and connecting the  $\mu\text{C}$  to a PicKit2 programmer [37], we read the fuses and observed that the read protection fuse for the code and internal EEPROM was enabled. Applying the methods of Section III, we tried to disable the readout protection with UV-C light.

To evaluate the vulnerability of the  $\mu\text{C}$  to the UV-C attack, we ordered a few blank PIC16F886 and decapsulated these test integrated circuits (ICs) with WFNA. Then, we took photographs of the silicon die of the  $\mu\text{C}$  with an optical microscope. An annotated image of the results is given in Fig. 6(a). To obtain an image of the area storing the fuse bits, the top layer of the silicon die was removed by sanding. The result shown in Fig. 6(b) reveals the same metal plates covering the configuration cells mentioned in [29]. However, instead of a “single line,” the row appears to be cut in half with one half being rotated by 180°, forming two “mirrored lines.”

We used an EPROM erasure tool as our UV-C source. After testing various positions and angles, we found that the configuration bits were erased after an exposure of about 20 min. Following Section III, the flash memory storing the program code and the internal EEPROM were protected from being deleted by covering the die with

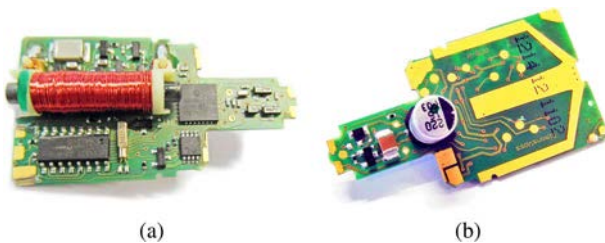


**Fig. 5.** PCB of door lock 3061. (a) Front. (b) Back.

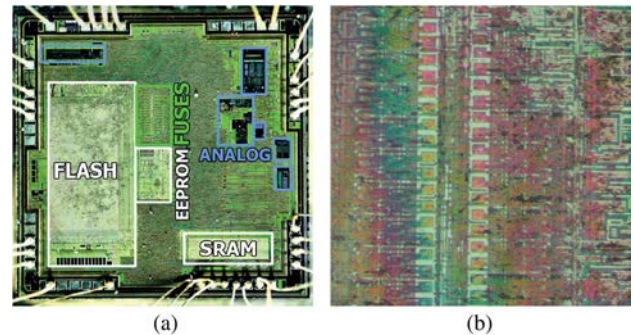


**Fig. 6.** Components of decapsulated PIC16F886. (a) PIC16F886 with memories (white), analog circuits (blue), and configuration bits (green) (analog to [30]). (b) Metal shielding over floating gates (top metal layer partially removed).

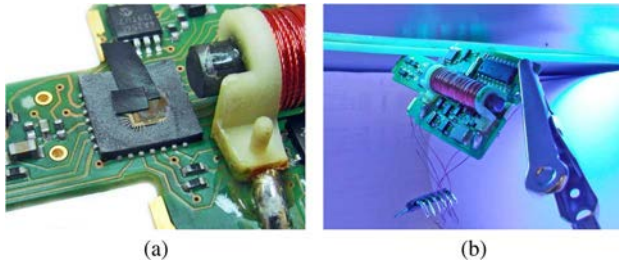


**Fig. 7.** Erasing fuses of the door's PIC16F886. (a) PCB after decapsulation, area around the fuses covered with electrical isolation tape. (b) PCB exposed to the light of a UV-C EPROM erasure tool.

electrical isolation tape, except for the location of the configuration bits. This resulted in only the fuse bits being reset, thus disabling the code readout protection. The contents of the program memory and the EEPROM remained fully intact. This procedure was successfully applied to several PIC16F886 taken from original SimonsVoss transponders and the respective door parts. As shown in Fig. 7, it is also possible to decapsulate the PIC16F886 directly on the PCB and read it out, with the electronics remaining fully functional.

In conclusion, we were able to recover the complete firmware (stored in the internal Flash memory) and the contents of the internal EEPROM of several transponders and door lock circuits.

## B. SimonsVoss Authentication Protocol

As a mandatory prerequisite for an SCA, we summarize the relevant results of reverse engineering the code running on the PIC of the transponder and the lock. To this end, we describe the key derivation mechanism, the cryptographic primitives, and the protocol used for mutual authentication.

The authentication protocol consists of 11 steps in total that are given in Fig. 8. In the symmetric-key scheme, the transponder and the lock prove that they know a shared long-term secret  $K_T$ . On each transponder, this individual 128-b key  $K_T$  is computed as the xor of a 128-b value  $K_{T,int}$  stored in the internal EEPROM of the  $\mu C$  (not accessible from the outside) and a 128-b value  $K_{T,ext}$  stored in the (unprotected) external EEPROM, i.e.,  $K_T = K_{T,ext} \oplus K_{T,int}$ . Each door within an entire SimonsVoss installation has an identical set of four 128-b keys  $K_{L,1}$ ,  $K_{L,2}$ ,  $K_{L,3}$ ,  $K_{L,4}$ , here called system key. Again, these keys are computed as the xor of values contained in the internal and external EEPROM. However, the internally stored value is identical for all four keys, i.e.,  $K_{L,j} = K_{L,j,ext} \oplus K_{L,int}$ . Note that when one of the four  $K_{L,j}$  has been recovered, the remaining three can be determined after reading the respective values from the external EEPROM.

The system key is used to derive the key  $K_T$  of a transponder to be authenticated. After receiving the identifier

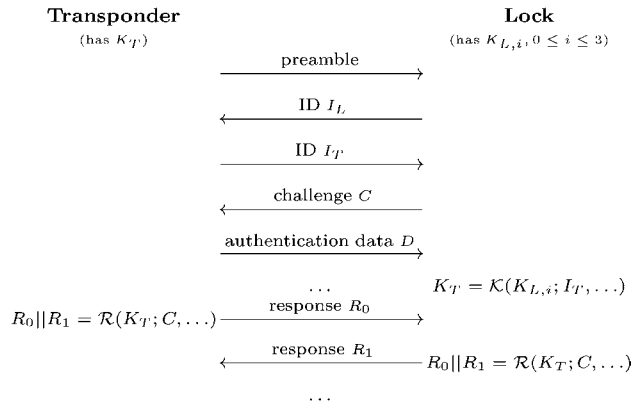


**Fig. 8.** Protocol for the mutual authentication between a transponder and a lock.

$I_T$  of a transponder, the lock computes  $K_T$  on the fly using a key derivation function  $\mathcal{K}$  involving the system key, previously exchanged “authentication data”  $D$ , and  $I_T$ . The authentication data  $D$  are a value sent by the transponder and control certain protocol functions.  $D$  is, however, transmitted in plaintext and unrelated to the cryptographic security of the protocol. All valid transponder identifiers that can authenticate to a particular door lock are stored unencrypted in its external EEPROM. Note that the key derivation is always executed by the door lock, even if it does not know the received  $I_T$ .

The key derivation  $\mathcal{K}$  consists of two building blocks: a modified DES denoted as  $\mathcal{D}$  and a SimonsVoss-proprietary function  $\mathcal{O}$  which we refer to as “obscurity function.”  $\mathcal{D}(K; M)$  is a slightly modified DES encrypting a 64-b plaintext  $M$  under a 64-b key  $K$ .

$\mathcal{O}(Y; X)$  consists of eight rounds  $r$ , with  $1 \leq r \leq 8$ . It takes a 16-B plaintext  $X = (x_0^{(1)} \dots x_{15}^{(1)})$  and a 16-B key  $Y = (y_0 \dots y_{15})$  to compute a 16-B ciphertext  $x_0^{(9)} \dots x_{15}^{(9)}$ . Fig. 9 depicts the structure of one round of  $\mathcal{O}$ . The internal “chaining values”  $c_1^{(r)} \dots c_{15}^{(r)}$  are processed horizontally in each round  $r$  and the last chaining value is used as the

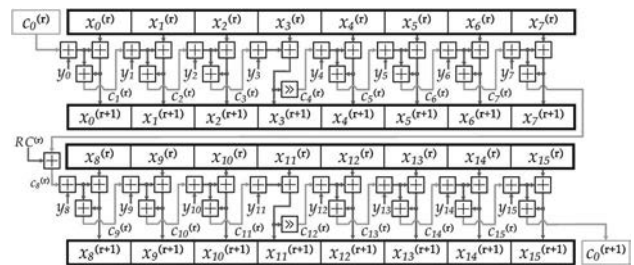


**Fig. 9.** One round  $r$  of the obscurity function  $\mathcal{O}$ : The key bytes  $y_i$  are constant, while the  $x_i^{(r)}$  and the chaining values  $c_i^{(r)}$  are updated in each round. The first chaining value is  $c_0^{(r)} = RC^{(0)}$ . The two 8-B halves  $x_0 \dots x_7$  and  $x_8 \dots x_{15}$  are processed identically.

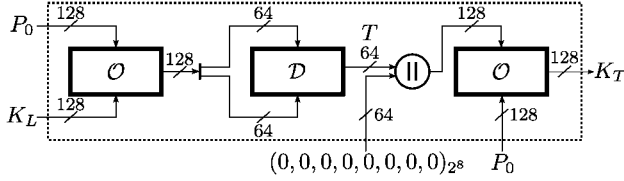


**Fig. 10. Construction of the key derivation function [7].**

input  $c_0^{(r+1)}$  for the subsequent round  $r + 1$ . The round constants  $RC^{(r)}$  are fixed byte values, i.e., they are identical for every execution of  $\mathcal{O}$ , that are added in each round after the first 8 B has been processed. The first chaining value is initialized with  $c_0^{(1)} = RC^{(0)}$ . All additions and shifts are performed modulo 256.

The key derivation function  $\mathcal{K}$  takes the system key and a 128-b parameter  $P_0$  as inputs, where  $P_0$  is derived from the first 3 B of  $I_T$  and the first 3 B of the authentication data  $D$  as

$$P_0 = (I_{T,0}, I_{T,1}, I_{T,2} \& 0x\text{C7}, D_0, D_1, D_2 \& 0x\text{3F}, 0, \dots, 0).$$

The transponder key  $K_T$  is then computed by a series connection of  $\mathcal{O}$ ,  $\mathcal{D}$ , and  $\mathcal{O}$ , as depicted in Fig. 10.

Depending on the two most significant bits (MSBs) of  $I_{T,2}$ , one of the four  $K_{L,j}$  is selected as the key for the first instance of  $\mathcal{O}$  in  $\mathcal{K}$  to encrypt  $P_0$ . The result is split into two 64-b halves, the lower 64 b being the plaintext and the upper 64 b being the key of  $\mathcal{D}$ . The 64-b result of  $\mathcal{D}$  is then padded with 64 zero bits and encrypted with  $\mathcal{O}$ , this time using  $P_0$  as the key. The resulting ciphertext is the transponder key  $K_T$  used in the subsequent steps of the challenge–response protocol. A lock is opened, if  $K_T$  on the transponder and  $K_T$  derived by the door match. To this end, both transponder and door compute a 64-b response involving the challenge  $C$  and several protocol values, as shown in Fig. 11, where

$$P_1 = (C_0, \dots, C_{10}, D_6, \dots, D_9, 0)$$

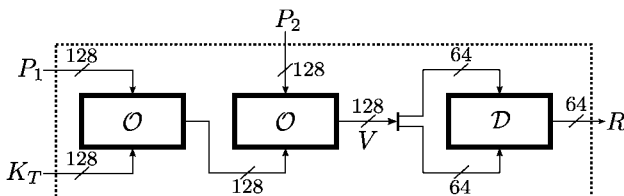


**Fig. 11. Construction of the response computation function [7].**

and

$$P_2 = (I_{L,2}, I_{T,2}, I_{T,3}, D_3, D_4, D_5, 0, \dots).$$

The transponder sends the first 32-b half  $R_0$  of the output of  $\mathcal{R}$ , to which the door responds with the second half  $R_1$ , if  $R_0$  was correct.

An analysis of the obscuring function  $\mathcal{O}$  has revealed serious weaknesses. Furthermore, it turned out that intermediate results of the response computation function were reused as part of the 11-B challenge message. Combining these two facts enables an attacker to compute a valid transponder key by only communicating with the digital cylinder. However, the manufacturer announced to fix this issue with a release of a firmware update. For more details, we refer to [7]. In the following, we focus on the susceptibility on SCA.

### C. Extraction of the System Key with SCA

Obviously, the key derivation function  $\mathcal{K}$  is the main target, because recovering the system key allows to derive the key of any transponder in an installation, given its  $I_T$ . In this part of the section, we describe the steps to perform a side-channel attack on the SimonsVoss door lock 3061. As the main result, we show that the system key can be extracted from the employed PIC  $\mu\text{C}$  with a noninvasive, CPA-based attack using approximately 150 traces. Possessing the system key, an adversary is able to create functionally identical clones of all transponders in an entire SimonsVoss installation. Note that the SCA can, in general, also be applied to a transponder, e.g., in order to duplicate it, but in a practical setting it is highly unlikely that an adversary will take the efforts for the attack just for cloning a single transponder.

1) *Predicting Intermediate Values in  $\mathcal{O}$* : The most promising target for an SCA to recover the system key is the first instance of  $\mathcal{O}$  in the key derivation. In the following, we (theoretically) derive an attack to obtain the full 16-B key  $y_0 \dots y_{15}$ .

Note that only the first 6 B of the input to  $\mathcal{O}$  can be chosen by the adversary, since the remaining bytes of  $P_0$  are set to zero. Due to the carry propagation properties of  $\mathcal{O}$  (which are described in detail in [7]), these input bytes do not lead to a “full” randomization of all intermediates in the first round of the obscuring function. A straightforward CPA of the first round targeting the addition operation  $x_i^{(r+1)} = x_i^{(r)} + c_i^{(r)} + y_i$  hence only allows to recover the first seven key bytes  $y_0 \dots y_6$ .

For the remaining key bytes, at least a part of the addition in the first round is carried out with completely constant data, ruling out a CPA: For revealing  $y_7$ , one obtains already two candidates, because the least significant bit (LSB) of  $c_7^{(1)}$  does not depend on the varying part

of the input and hence remains constant. For  $c_8^{(1)}$ , two bits are not randomized, leading to four candidates for  $y_8$  (if  $c_7$  was known). Thus, to obtain these two bytes, two CPAs, each with four additional candidates, would be necessary. To recover all key bytes using this approach, an overall number of  $2^{1+2+3+4+5+4+5+6+7} = 2^{37}$  key candidates would have to be tested, leading to an impractical attack. Hence, we utilize further properties of  $\mathcal{O}$  to reduce the computational cost by extending the attack to initially nonrecoverable key bytes in subsequent rounds of  $\mathcal{O}$ .

First, note that all bits in the (initially not fully randomized) update involving the key bytes  $y_7 \dots y_{15}$  are fully dependent on  $x_0^{(1)} \dots x_5^{(1)}$  after two, three, four, five, six, six, seven, and seven rounds, respectively. Thus, these key bytes can be recovered by means of a CPA in the respective round, if all other values preceding the update operation for a targeted key byte  $y_i$  can be simulated. Assuming that all key bytes up to  $y_i$  and all  $c_0^{(r)}$  up to the respective round are known, the only unknown constant in the  $i$ th update operation is  $y_i$ . The correct value can be determined using a CPA with 256 candidates for  $y_i$  in the first round, enabling to calculate the updated  $y_i$  in the targeted round.

The remaining problem with this attack is how to determine  $c_0^{(r)}$ . For this, note that in the second round,  $c_0^{(2)}$  is independent of  $x_0^{(1)} \dots x_5^{(1)}$ , i.e., a constant only depending on the (constant) key bytes  $y_i$ . Hence,  $c_0^{(2)}$  can be found using a CPA, because  $y_0$  was already determined in the first round, so  $x_0^{(2)}$  can be computed. The CPA for obtaining  $c_0^{(2)}$  is performed with 256 candidates. Having found  $c_0^{(2)}$ , all values up to the update of  $x_7^{(2)}$  can be computed. This update, in turn, only depends on known values ( $x_7^{(1)}$ ,  $c_7^{(1)}$ , and  $c_7^{(2)}$ ) and the unknown key byte  $y_7$ . Hence, with 256 candidates,  $y_7$  can be determined with a CPA.

In the third, fourth, and fifth round, only the MSBs of  $c_0^{(3,4,5)}$  vary. The remaining bits are constant and can be recovered. In addition, due to the multiplication by two (i.e., a binary left shift) in the propagation of  $c_i$ , the unknown MSB only affects (the MSB of)  $x_0^{(4)}$ ,  $x_0^{(5)}$ ,  $x_1^{(5)}$ ,  $x_0^{(6)}$ ,  $x_1^{(6)}$ , and  $x_2^{(6)}$ . Subsequent bytes do not depend on the unknown MSB and can be fully predicted. Hence, it is possible to recover  $y_8$ ,  $y_9$ , and  $y_{10}$  in rounds three to five. Finally, in the sixth, seventh, and eighth round, the three MSBs of  $c_0^{(6)}$  and the five MSBs of  $c_0^{(7)}$  and  $c_0^{(8)}$  vary. Again, the constant part of  $c_0^{(6,7,8)}$  can still be recovered. For the varying three MSBs, only  $x_0^{(7)} \dots x_2^{(7)}$  are affected. This recoverable part is sufficient to fully predict the state update for the targeted key bytes  $y_{11}$ ,  $y_{12}$ , and  $y_{13}$ . Finally, for round seven, the change in the five MSBs of  $c_0^{(7)}$  only affects  $x_0^{(8)} \dots x_6^{(8)}$ , posing no problem to the recovery of the remaining bytes  $y_{14}$  and  $y_{15}$ .

In short, the attack can be summarized as follows: First, one recovers the first seven key bytes in round one. Then, the constant part of  $c_0$  at the beginning of each

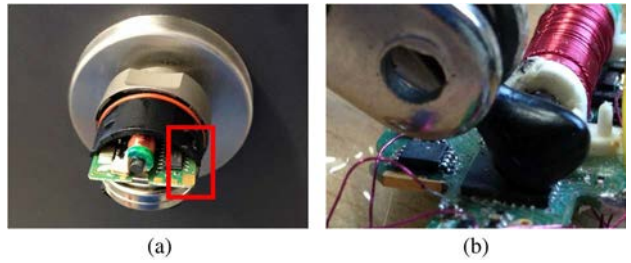


**Fig. 12.** Setup and EM trace for SCA of the door part. (a) PCB in the door part. (b) EM probe on PIC.

round has to be found. Finally, all key bytes for which the update operation is fully randomized in the respective round can be obtained.

2) *Practical Results:* In the following, we first describe the measurement setup used to acquire side-channel traces for the SimonsVoss door part 3061. By applying the attack described in Section V-C1 to real-world power traces, we recover the system key using a limited number of power traces in a noninvasive manner.

To trigger the key derivation on the DUT, we directly connect to the data lines between the ASIC and the PIC. The respective pins [marked with a red rectangle in Fig. 12(a)] are accessible without removing the PCB from the door. Equivalently, the communication with the DUT could also be performed sending data over the RF interface, e.g., using the USRP2. We primarily decided not to use the RF interface to increase the (mechanical) stability of the measurement setup.

We noninvasively measured the power consumption during the execution of the key derivation function for randomly chosen  $ID_T$  and authentication data  $D$  (apart from a few bits that are required to be set by the protocol). To this end, we placed an EM probe on the package close to the power supply pins of the PIC; cf., Fig. 12(b). Initial experiments to measure the power consumption by inserting a shunt resistor into the battery connection yielded heavily smoothed power traces, presumably due to several bypass/voltage stabilization capacitors on the PCB. Moreover, the ASIC is also connected to the main battery supply, resulting in wideband, high-amplitude noise that could not be filtered out.

In contrast, the EM probe at the given position mainly picked up the power consumption of the PIC. Using the described setup, we recorded 1000 power traces using a digital storage oscilloscope at a sample rate of 500 MHz. With the current measurement setup, due to delays caused by the protocol, about ten traces can be acquired per minute. Note that the PIC runs on the internal RC oscillator at a frequency of approximately 8 MHz. Hence, we further (digitally) bandpass-filtered the recorded traces. Experimentally varying the lower and upper frequency of

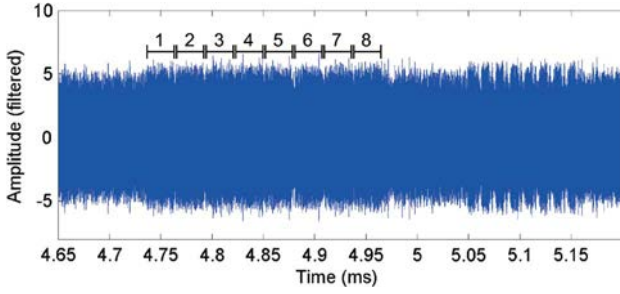


**Fig. 13.** Filtered EM trace during the execution of the first obscurity function in the key derivation.

the passband, we found that a passband from 6 to 9 MHz yields the best result.

To determine the relevant part of the power trace belonging to the key derivation, we initially inserted a small function into the (otherwise unmodified) code of the PIC using the method described in Section V-A. This function generates a rising edge on an unused pin of the PIC, serving as a trigger signal for profiling purposes. Fig. 13 exemplarily depicts the part of a trace belonging to the initial execution of  $\mathcal{O}$  in the key derivation. The eight rounds of the function can be recognized as eight distinct “humps.” Furthermore, a unique pattern occurs at the beginning of the relevant part (and each round). This pattern can serve for alignment purposes: Having profiled the DUT, we removed the artificial trigger signal and recorded traces for the original, unmodified code. We then used the pattern to align the traces for the actual CPA attack.

Having determined the relevant part of the trace and the appropriate preprocessing steps, we practically performed the (theoretical) attack described in Section V-C1. We use the HD between a byte  $x_i^{(r)}$  and its updated value  $x_i^{(r+1)}$  as the power model, i.e.,  $h = \text{HD}(x_i^{(r)}, x_i^{(r+1)})$  (dropping the indices  $n$  for the trace and  $k$  for the key candidate for better readability). This model was derived based on the analysis of the assembly code implementing  $\mathcal{O}$  and the leakage model for the PIC series described in [31]. Using this model, we obtained correlation values of approximately 0.75 for the correct candidate, while all other (wrong) candidates exhibit a lower value. This allows to unambiguously determining the key with approximately 100 traces, as exemplarily depicted in Fig. 14(a).

The CPAs for  $c_0^{(r)}$  exhibit a similar behavior; cf., for instance, Fig. 14(b) for  $c_0^{(2)}$ . However, for determining the partial  $c_0^{(r)}$  in later rounds, a (slightly) higher number of traces is needed (especially for round six and seven), since only the LSBs of the respective intermediate values are predicted correctly. Still, approximately 150 traces are sufficient to obtain the maximum correlation at the correct point in time for the correct candidate.

Note that, at a higher computational cost, the CPA on  $c_0^{(6)}$  and  $c_0^{(7)}$  could be left out altogether: The attack on the

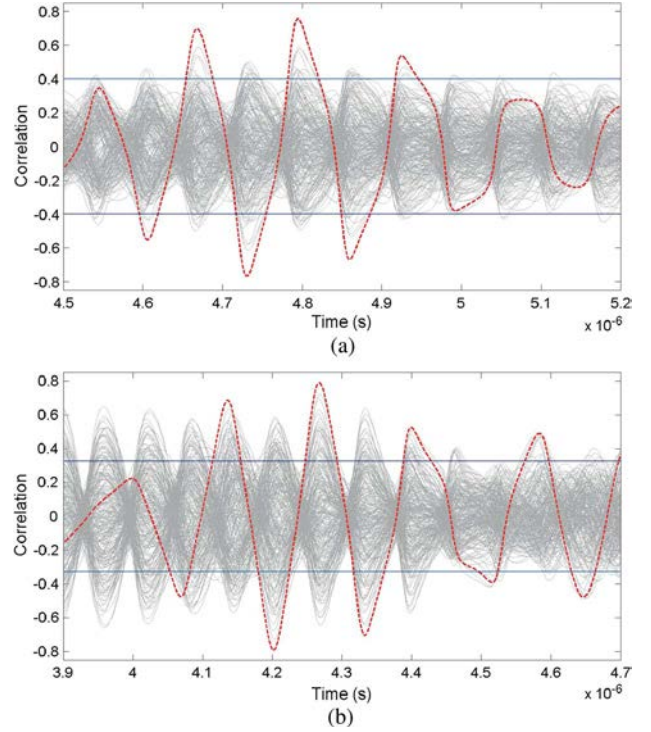


**Fig. 14.** Correlation for key byte  $y$ , after 100 traces and  $c_0^{(2)}$  after 150 traces. Correct candidate: red, dashed. (a)  $y$ , in round 2. (b)  $c_0^{(2)}$  in round 2.

actual key bytes in rounds six and seven could be carried out for all  $2^5$  or  $2^3$  candidates for  $c_0^{(6)}$  or  $c_0^{(7)}$ , respectively. This would increase the number of candidates to  $2^5 \cdot 2^8 = 2^{13}$  and  $2^3 \cdot 2^8 = 2^{11}$ . For this amount of candidates, a CPA can still be executed efficiently.

## VI. CASE STUDY II: THE YUBIKEY ONETIME PASSWORD TOKEN

As the second case study, we focus on a  $\mu\text{C}$ -based OTP token, the YubiKey 2 by Yubico [11]. In general, OTP tokens are used in scenarios where the classical way of authentication with a username/password pair is insufficient. For example, considering remote login into a corporate network, an adversary can choose from a multitude of methods to obtain the credentials, among others, by guessing passwords using a dictionary or by installing malware on the system of the target user.

The YubiKey employs an open-source protocol based on the mathematically secure AES. By emulating a USB keyboard, entering the OTP is platform independent. The token itself comprises a secret cryptographic key that, together with timestamps and counters, is used to derive a fresh OTP for each authentication. Yubico has several security-sensitive reference customers (that use the YubiKey, e.g., for securing a remote access) listed on their

website [11], e.g., Novartis, Agfa, and the U.S. Department of Defense Contractors. The U.S. Department of Defense Contractors even switched from RSA's SecureID system to the YubiKey [38], even though the YubiKey 2 is not certified for governmental standards.

### A. Two-Factor Authentication

The OTP generated by the YubiKey 2 is provided in addition to the normal credentials and is only valid for a single use. Hence, the user has to know the username and password and additionally has to own the token to successfully perform an authentication. The OTP generated by the YubiKey 2 is based on several counters, random bytes, a secret ID, and a checksum, which are concatenated to a 16-B value and subsequently encrypted using the AES with a 128-b key. The exact structure of the OTP is as follows:

- **UID:** The private ID is 6 B long and kept secret. It can be used as another secret parameter or to distinguish users when a common encryption key is used.
- **useCtr:** The usage counter is 2 B long and increased at the first OTP generation after power-up. Additionally, the counter is incremented when the session usage counter wraps from  $0\text{x}\text{ff}$  to  $0\text{x}00$ .
- **tstp:** The 3-B timestamp is initialized with random data on startup and is incremented approximately eight times per second.
- **sessionCtr:** The 1-B session counter is volatile, starts with  $0\text{x}00$  at power-up and is incremented on every OTP generation.
- **rnd:** Two additional bytes of random data.
- **crc:** A 2-B CRC16 checksum computed over all previous data fields.

### B. Hardware of the YubiKey 2

The YubiKey 2 is mono-block molded and thus hermetically sealed. To find out which kind of  $\mu\text{C}$  is used in the YubiKey 2, we dissolved the casing with WFNA to gain access to the silicon die [cf., Fig. 15(a)]. The position of the  $\mu\text{C}$  was known from a promotional video about the production of the YubiKey [39] from which we extracted the picture of the PCB shown in Fig. 15(a). On the die, we found the label "SUNPLUSIT" [cf., Fig. 15(c)] which seems to belong to Sunplus Innovation Technology Inc. based in Taiwan [40]. We were unable to exactly find out which controller was used, as there is no Sunplus part related to the label "AV7011." However, all human interface device (HID)  $\mu\text{Cs}$  produced by Sunplus employ an 8-b architecture.

### C. Measurement Setup

To record power traces for an SCA, we built a simple adapter to get access to the USB power and data lines; cf., Fig. 16(a). The D+ and D- lines are directly connected to

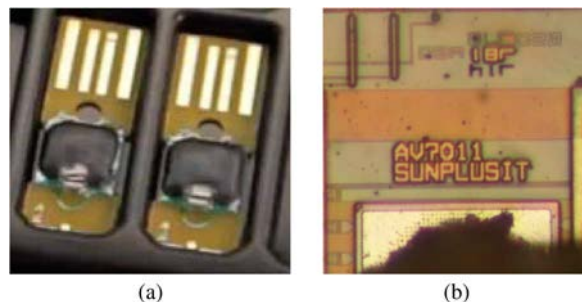


**Fig. 15.** YubiKey 2 during packaging and label on the silicon die. (a) PCB (source: [39]). (b) Die label.

the PC's USB port. A  $60\ \Omega$  resistor was inserted into the ground line to measure the power consumption of the YubiKey 2.

In our first experiments, we used  $V_{\text{CC}}$  provided by the USB port as power supply for the YubiKey, however, this resulted in a high amount of measurement noise. Therefore, an external power supply was added to reduce the noise caused by the PC's power supply. A custom amplifier was included to amplify the measured voltage drop over the resistor. All measurements were recorded at a sample rate of 500 MHz. Fig. 17(a) depicts the overall structure of the setup.

Initially, to perform the profiling of the DUT described in Section VI-D, we focused on the power consumption measured via the shunt resistor. However, in subsequent experiments and for improving the key recovery described in Section VI-E, we also recorded the EM emanation of the DUT by placing a commercially available near-field probe [41] on the position on the package of the YubiKey 2 with the highest signal amplitude. The resulting signal was amplified by 30 dB using an amplifier made by Langer EMV [42]. The EM probe on the casing to the YubiKey 2 is depicted in Fig. 16(b).

### D. Side-Channel Profiling

Initially, examining the power trace of the DUT, there was a significant voltage drop visible caused by the

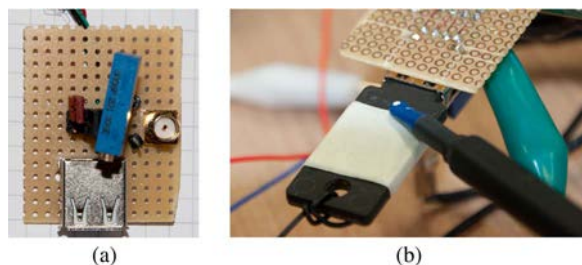


**Fig. 16.** Measurement methods: USB adapter with shunt resistor and EM probe at the position with maximal signal amplitude on the YubiKey 2. (a) USB adapter. (b) EM probe.

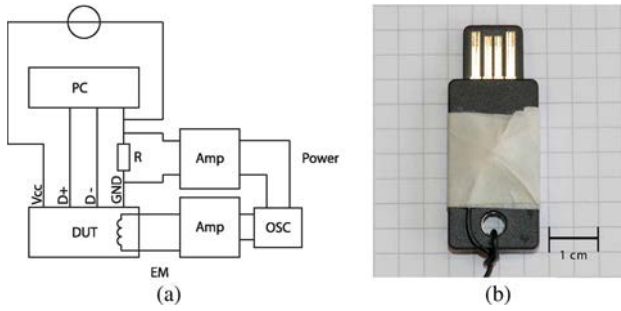


**Fig. 17.** Setup for measuring the power consumption and EM emanation. (a) Schematic. (b) YubiKey 2 with wire for simulating button presses.

YubiKey turning off its LED. This point was used as a reference point to trigger the data acquisition of the oscilloscope. Note that at least 2.6 s are required to “recover” after a button press. Incidentally, this significantly slows down the measurement process (and thus the overall attack) because the speed of the data acquisition is limited by this property of the YubiKey.

Right before the voltage drop caused by the LED, a pattern can be observed that resembles a structure with ten rounds, each approximately 200  $\mu$ s long. Since the AES-128 employed on the YubiKey has ten rounds, it is likely that this part of the trace belongs to the AES encryption. This is further confirmed by Fig. 18 showing an average trace computed using 1000 power traces. The “rounds” are clearly visible, and even different operations are distinguishable within one round. Note that, however, we were unable to observe single instructions within one round, rather, it appears the traces are in some way low-pass filtered. This might be due to a voltage regulator of the  $\mu$ C or decoupling capacitors. Additionally, the tenth round at approximately 2.1 ms is 70  $\mu$ s shorter than the others, which agrees with the fact that the final round of the AES algorithm misses the MixColumns step.

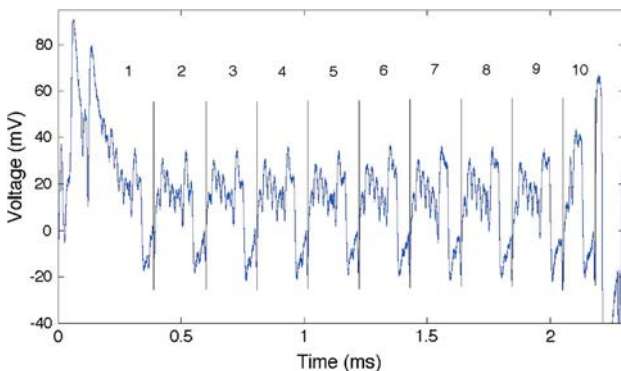


**Fig. 18.** Average over 1000 amplified traces of the part suspected to belong to the AES encryption.

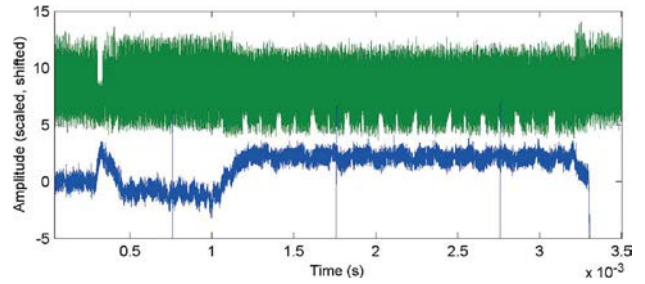


**Fig. 19.** Power consumption trace (blue, bottom) and demodulated EM trace (green, top). Vertical scaling and offset changed to compare general signal shape.

We recorded 20 000 traces of the part presumably belonging to the AES operation. The 128-b AES key was set to a fixed value. The used sample rate was 500 MHz, as mentioned in Section VI-C. Experimentally, we found that (digitally) downsampling the traces by a factor of ten does not affect the success rate of the subsequent attack presented in Section VI-E. Hence, to reduce the data and computation complexity, all experiments described in the following included this preprocessing step. We tested different models for the power consumption of the device. An 8-b HW model for single bytes of the intermediate values within the AES turned out to be suitable, confirming the assumption that an 8-b  $\mu$ C is used in the YubiKey 2.

As mentioned in Section VI-C, we also captured the EM emanation of the DUT at the same time as the power consumption in subsequent experiments. The EM traces mainly showed a clock signal at a frequency of 12 MHz. However, digitally amplitude demodulating [43] this signal yielded a trace not exhibiting the low-pass-filtered shape observed for the power consumption traces. Fig. 19 depicts a power consumption trace (blue, bottom) and the corresponding demodulated EM trace (green, top). In both cases, the round structure is discernible. Yet, the EM trace allows to separately observe every clock cycle, while the power consumption trace only shows the overall round structure.

Similar to the power consumption traces, we also observed distorted EM traces. However, the overall number of “usable” traces was higher compared to the power consumption measurement: Only 25% of the EM traces had to be discarded, compared to about 35% of the traces acquired with the shunt.

### E. Practical Attack: Extracting the AES Key

Having analyzed the round structure and identified the points in time when the leakage occurs, we continued with trying to recover the secret AES key. Initially, we used the power traces, but switched to EM traces later to reduce the number of required measurements and thus the time needed for the attack.

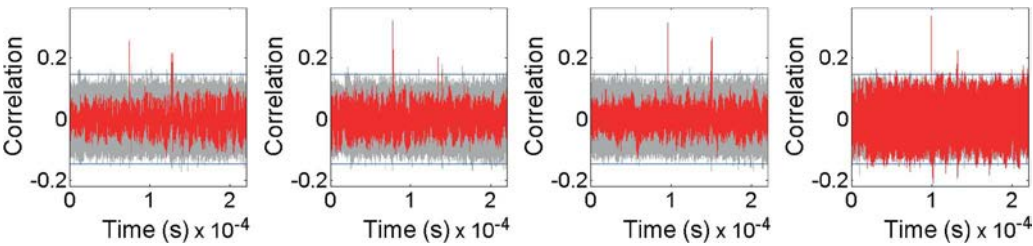


**Fig. 20.** Correlation coefficient for all candidates for the key bytes 1, 2, 8, and 9 (left to right) after 800 traces. Red: correct key candidate; gray: wrong key candidates.

We computed the correlation coefficient for all 256 candidates for each key byte using 10 000 power traces. The hypothetical power consumption  $h_i$  (cf., Section II-C) was computed as  $h_i = HW(SBOX^{-1}(C_i \oplus rk))$ , with  $C_i$  being a ciphertext byte (for measurement  $i$ ) and  $rk$  the corresponding byte of the round key (dropping the byte index for better readability).

We were able to clearly determine the full 128-b AES key using approximately 4500 traces. In this regard, it turned out that the number of traces needed to recover a key byte differs: For byte 1, 4, and 16, less than 1000 traces were sufficient. For byte 8, 9, 10, 11, 13, and 14, less than 3000 traces sufficed to determine the correct value. For byte 2, 3, 5, 6, 7, 12, and 15, a number between 3100 and 4500 traces leads to the correct key byte being found. Note that, due to the glitches mentioned in Section VI-D, the necessary preselection of the traces effectively requires more traces to be recorded: For 4500 usable traces, approximately 7000 traces had to be measured in total. With our current measurement setup, 1000 traces can be acquired in about 1.5 h, corresponding to a rate of 11.1 traces/min. Thus, to obtain 7000 traces in total, approximately 10.5 h of access to the DUT were necessary.

We observed a “spread” correlation peak with a width of 8.3  $\mu$ s, which would translate to a clock frequency of approximately 120 kHz. At this clock frequency, the execution time of about 2.5 ms (cf., Fig. 18) would imply that the AES is performed in only 300 clock cycles. Considering that even highly optimized AES implementations require about 3000 cycles on similar (and probably more powerful) 8-b  $\mu$ Cs [44], it appears that the leakage is distributed over several clock cycles, presumably due to the low-pass characteristic mentioned in Section VI-D. Hence, we continued our analysis using the EM traces that give a higher resolution in this regard.

To this end, we performed the identical attack on the (digitally demodulated) EM traces. The resulting correlations after 800 traces for all candidates for the first, second, eighth, and ninth key bytes are exemplarily shown in Fig. 20. In contrast to the power consumption traces, the correlation for the correct key candidate clearly exceeds the one for the wrong candidate after already less than 1000 traces. Besides, the correlation peak is limited to

a short instant of approximately 160 ns, which corresponds to a clock frequency of about 6.25 MHz. Thus, it is likely that this correlation is for one or a few instructions of the  $\mu$ C only.

To get an estimate of how many traces are needed to clearly distinguish the correct key candidate from the wrong ones, the maximum correlation coefficient (at the point of leakage) for each candidate after each trace was saved. We used the ratio between the maximum correlation for the correct key candidate and the highest correlation for the “second best” wrong candidate as a metric; cf., for instance, [45]. We then took the number of traces for which this ratio is greater than 1.1 as the minimum number of required traces. The results are given in Table 2: 500 traces are sufficient to fully recover the 128-b AES key. Due to approximately 25% of the EM traces being unusable, this translates to an overall number of around 650 traces. Thus, only 1 h of access to the YubiKey 2 is sufficient to recover the key with EM measurements, compared to 10.5 h that would be required when using the power consumption traces. Besides, a tradeoff between computation time and the number of traces could be applied. An adversary may, for instance, decide to only record 300 traces, so three key bytes (1, 3, and 14) would not be (fully) recoverable. However, these remaining three bytes, i.e., 24 b, could be easily determined using an exhaustive search on a standard PC within minutes. In this case, the measurement time is reduced to 36 min for effective 400 traces in total.

**Table 2** Approximate Number of Required Traces to Recover Respective Bytes of the AES Key Using EM Traces. Metric: Ratio Between Correlation for Correct Key Candidate and Second Highest Correlation Greater Than 1.1

| Key byte          | 1   | 2   | 3   | 4   |
|-------------------|-----|-----|-----|-----|
| # Required traces | 400 | 300 | 400 | 200 |
| Key byte          | 5   | 6   | 7   | 8   |
| # Required traces | 300 | 200 | 300 | 200 |
| Key byte          | 9   | 10  | 11  | 12  |
| # Required traces | 200 | 200 | 200 | 200 |
| Key byte          | 13  | 14  | 15  | 16  |
| # Required traces | 300 | 500 | 300 | 300 |

Given the AES key, an adversary is able to generate an arbitrary number of valid OTPs and thus to impersonate the legitimate owner given that the traditional credentials have been obtained, e.g., by means of eavesdropping, phishing, or malware. Note that a standard CPA was sufficient to mount our attack with a number of traces small enough to pose a threat in the real world. Hence, we did not further investigate more complicated (profiled) SCA techniques like TAs, as described in Section II.

Further, the here presented attack leaves no physical traces on the DUT. The only means by which the attack could be detected is a (relatively high) increase of the usage counters (cf., Section VI-A). Due to the fact that the volatile session counter has to reach 256 first before the nonvolatile usage counter is incremented, the EM-based attack only increases the usage counter by two when recording 500 traces. Thus, the presented attack does not lead to a “suspicious” change of this counter and is very unlikely to be detected in this way.

## VII. CONCLUSION

### A. Summary

In this paper, we demonstrated that standard  $\mu$ Cs are vulnerable to numerous attacks, including the extraction of the embedded code (Sections III and IV) and of cryptographic secrets by means of SCA (Section II). Consequently, using off-the-shelf  $\mu$ Cs for applications with moderate to high security requirements may render the resulting system insecure, even if the employed cryptographic primitives (e.g., AES) are secure from a mathematical point of view.

The case studies of Sections V and VI give examples in this regard: In both cases, implementation attacks could be used to extract cryptographic keys, giving an adversary access to secured buildings (Section V) or, e.g., protected web services and networks (Section VI), respectively. Note that the susceptibility of real-world devices toward SCA and related techniques is a general problem—further examples for vulnerable devices include KeeLoq door openers [13], secure EEPROMs made by Atmel [46], Mifare DESFire MF3ICD40 contactless smartcards [15], and FPGAs by Xilinx, Actel, and Altera [47]–[50].

Nevertheless, in terms of security, standard  $\mu$ Cs mark the lower end: While the attacks presented in this paper work with less than 150 (SimonsVoss lock) and 700 traces (YubiKey 2), respectively, SCA of more secure (hardware) implementations may require several tens or hundred thousand traces to succeed [15], [50]. Moreover, reverse-engineering vendor-specific functionality by reading out embedded code is usually easier (in terms of required time, knowledge, and equipment) than analyzing FPGA configuration bitstreams or directly the silicon die of ICs.

Finally, we would like to stress that SimonsVoss and Yubico had been notified well in advance of the disclosure

of the respective weaknesses. To our knowledge, both companies attempt to improve the security of their products.

### B. Countermeasures and Future Work

As a major point for future work, the question arises how to protect embedded (cryptographic) systems against the presented attack methods. With respect to SCA, various countermeasures have been proposed, with an overview given in [3]: As a first step, one may attempt to eliminate the leakage on the level of the circuit itself. To this end, custom logic styles to balance the power consumption have been designed; cf., for example, [51]. However, in practice, it turns out that completely removing the side-channel leakage is nearly impossible, e.g., due to manufacturing tolerances and so on. Thus, SCA-based attacks still apply, but may require a higher number of traces. The same holds for countermeasures based on generating additional noise, e.g., by randomly switching a large load in parallel to the execution of the cryptographic operation. In general, balancing the power consumption reduces the signal-to-noise ratio (SNR) (decreasing the leakage signal), while the generation of noise achieves the same by increasing the amplitude of the noise.

In addition to focusing on the amplitude dimension, the time dimension can also be used to impede SCA. Varying the clock signal or inserting “dummy” operations leads to a desynchronization of the traces, increasing SNR and thus increasing the amount of traces needed for an attack. On the algorithmic level, the order of certain (independent) operations (e.g., S-Box lookups in a block cipher) may be randomized, a technique referred to as shuffling in the literature.

Finally, the intermediate values within the state of the algorithm itself can be randomized by adding a so-called “mask” that varies for each execution of the algorithm. Such masking schemes [52], [53] remove the dependency between a predicted intermediate and the observed side-channel signal altogether, however, often come with significant (time and/or space) overhead and thus can be too costly for resource-constrained devices. Thus, although numerous countermeasures against implementation attacks and SCA in particular have been proposed in the literature, further investigations in this area are of interest, especially when it comes to protecting low-cost and thus significantly constrained devices such as  $\mu$ Cs.

With respect to the extraction of the embedded code of  $\mu$ Cs, both offensive and defensive aspects deserve further attention. Most countermeasures available today are designed to prevent highly invasive attacks: For example, (active) meshes cover the silicon die and the data bus itself is encrypted to impede microprobing. Nevertheless, even high-security devices protected with such countermeasures can still be attacked, as, for example, demonstrated for the Infineon SLE 66PE that was successfully attacked by Tarnovsky [54] using a focused ion beam (FIB).

Moreover, new methods like side-channel code extraction (cf., Section IV) or the use of the photonic side channel [55] (which gives a spatially resolved map of the activity of an IC) have likely not been considered in the de-

velopment process of secure ICs so far. Devising at the same time effective and cost-efficient countermeasures against these threats is hence a highly relevant area for future research. ■

## REFERENCES

- [1] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, other systems," *CRYPTO 1996*, vol. 1109, N. Kobitz, Ed. Berlin, Germany: Springer-Verlag, 1996, pp. 104–113, ser. Lecture Notes in Computer Science.
- [2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *CRYPTO 1999*, vol. 1666, M. J. Wiener, Ed. Berlin, Germany: Springer-Verlag, 1999, pp. 388–397, ser. Lecture Notes in Computer Science.
- [3] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. New York, NY, USA: Springer-Verlag, 2007.
- [4] S. P. Skorobogatov, "Semi-invasive attacks—A new approach to hardware security analysis," *Comput. Lab., Univ. Cambridge, Cambridge, U.K., Tech. Rep. UCAM-CL-TR-630*. [Online]. Available: <http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-630.pdf>
- [5] T. Eisenbarth, C. Paar, and B. Weghenkel, "Building a side channel based disassembler," *Trans. Comput. Sci.*, vol. 10, pp. 78–99, 2010.
- [6] D. Malysiak, "GPU assisted implementation of kernel-based template attacks," M.S. thesis, Horst Görtz Institute for IT-Security, Ruhr Univ. Bochum, Bochum, Germany, 2011. [Online]. Available: [http://www.emsec.rub.de/media/crypto/attachments/files/2011/08/Thesis\\_Malysiak\\_Darius.pdf](http://www.emsec.rub.de/media/crypto/attachments/files/2011/08/Thesis_Malysiak_Darius.pdf)
- [7] D. Strobel et al., "Fuming acid and cryptanalysis: Handy tools for overcoming a digital locking and access control system," *CRYPTO 2013*, vol. 8042, R. Canetti and J. A. Garay, Eds. Berlin, Germany: Springer-Verlag, 2013, pp. 147–164, ser. Lecture Notes in Computer Science.
- [8] Industrytoday, "SimonsVoss leads European access control market," 2010. [Online]. Available: <http://www.industrytoday.co.uk/security/simonsvoss-leads-european-access-control-market/2227>
- [9] SimonsVoss Technologies AG, "The finest in keyless security," 2013. [Online]. Available: [http://www.simons-voss.de/fileadmin/media/Imagebrotschueren/SimonsVoss\\_I\\_mage\\_EN.pdf](http://www.simons-voss.de/fileadmin/media/Imagebrotschueren/SimonsVoss_I_mage_EN.pdf)
- [10] D. Oswald, D. Strobel, F. Schellenberg, T. Kasper, and C. Paar, "When reverse-engineering meets side-channel analysis—Digital lockpicking in practice," *Selected Areas in Cryptography*. Berlin, Germany: Springer-Verlag, 2013, pp. 571–588.
- [11] Yubico, "Two-factor authentication with the YubiKey," 2013. [Online]. Available: <http://www.yubico.com>
- [12] D. Oswald, B. Richter, and C. Paar, "Side-channel attacks on the Yubikey 2 one-time password generator," *Research in Attacks, Intrusions, Defenses*, vol. 8145, S. J. Stolfo, A. Stavrou, and C. V. Wright, Eds. Berlin, Germany: Springer-Verlag, 2013, pp. 204–222, ser. Lecture Notes in Computer Science.
- [13] T. Eisenbarth et al., "On the power of power analysis in the real world: A complete break of the KeeLoq code hopping scheme," *CRYPTO 2008*, vol. 5157. Berlin, Germany: Springer-Verlag, 2008, pp. 203–220, ser. Lecture Notes in Computer Science.
- [14] T. Kasper, M. Silbermann, and C. Paar, "All you can eat or breaking a real-world contactless payment system," *Financial Cryptography 2010*, vol. 6052. Berlin, Germany: Springer-Verlag, 2010, pp. 343–350, ser. Lecture Notes in Computer Science.
- [15] D. Oswald and C. Paar, "Breaking Mifare DESFire MF3ICD40: Power analysis and templates in the real world," *CHES 2011*, vol. 6917. Berlin, Germany: Springer-Verlag, 2011, pp. 207–222, ser. Lecture Notes in Computer Science.
- [16] S. Mangard, "A simple power-analysis (SPA) attack on implementations of the AES key expansion," *ICISC 2002*, vol. 2587. Berlin, Germany: Springer-Verlag, 2002, pp. 343–358, ser. Lecture Notes in Computer Science.
- [17] C. D. Walter, "Simple power analysis of unified code for ECC double and add," *CHES 2004*, vol. 3156, M. Joye and J.-J. Quisquater, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 191–204, ser. Lecture Notes in Computer Science.
- [18] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," *CHES 2004*, vol. 3156, M. Joye and J.-J. Quisquater, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 16–29, ser. Lecture Notes in Computer Science.
- [19] E. W. Weisstein, "Variance," *Mathworld—A Wolfram Web Resource*. Dec. 2010 [Online]. Available: <http://mathworld.wolfram.com/Variance.html>
- [20] J.-L. Danger et al., "DPA contest v2: Rules of the contest." [Online]. Available: <http://www.dpacontest.org/v2/rules.php>
- [21] Research Center for Information Security, National Institute of Advanced Industrial Science and Technology, "Side-channel attack standard evaluation board SASEBO-GII specification," 2009. [Online]. Available: [http://www.dpacontest.org/v2/hall\\_of\\_fame.php](http://www.dpacontest.org/v2/hall_of_fame.php)
- [22] DPA Contest v2, "Hall of Fame." [Online]. Available: [http://www.dpacontest.org/v2/hall\\_of\\_fame.php](http://www.dpacontest.org/v2/hall_of_fame.php)
- [23] Texas Instruments Inc., "MSP430 Programming Via the JTAG Interface—Users Guide," 2013. [Online]. Available: <http://www.ti.com/lit/ug/slau320k/slau320k.pdf>
- [24] Infineon Technologies AG, "TC1797 32-bit Single-Chip Microcontroller Data Sheet V1.2 2009-09," 2009.
- [25] M. Meriac, "Heart of darkness—Exploring the uncharted backwaters of HID iCLASS security," in *Proc. 27th Chaos Commun. Congr.*, 2010. [Online]. Available: [http://www.openpcd.org/HID\\_iClass\\_demystified](http://www.openpcd.org/HID_iClass_demystified)
- [26] Microchip Technology Inc., "Microchip technology delivers 12 billionth PIC microcontroller to leading motor manufacturer," Nidec Corporation, 2013. [Online]. Available: <http://www.microchip.com/pagehandler/en-us/press-release/microchips-12-billionth-pic-mi.html>
- [27] Microchip Technology Inc., "Investor Presentation," 2013. [Online]. Available: [http://www.microchip.com/investor/Pressrelease/MCHP\\_tation.060313.pdf](http://www.microchip.com/investor/Pressrelease/MCHP_tation.060313.pdf)
- [28] IC INSIGHTS, INC., "MCU market on migration path to 32-bit and ARM-based devices," 2013. [Online]. Available: <http://www.icinsights.com/data/articles/documents/541.pdf>
- [29] A. Huang, "Hacking the PIC 18F1320," 2005. [Online]. Available: [http://www.bunniestudios.com/blog/?page\\_id=40](http://www.bunniestudios.com/blog/?page_id=40)
- [30] A. Zonenberg, "Microchip PIC12F683 Teardown," 2011. [Online]. Available: <http://siliconexposed.blogspot.de/2011/03/microchip-pic12f683-teardown.html>
- [31] M. Goldack, "Side-channel based reverse engineering for microcontrollers," Diploma thesis, Horst Görtz Institute for IT-Security, Ruhr University Bochum, Bochum, Germany, 2008. [Online]. Available: [https://www.emsec.rub.de/media/attachments/files/2012/10/da\\_goldack.pdf](https://www.emsec.rub.de/media/attachments/files/2012/10/da_goldack.pdf)
- [32] F.-X. Standaert and C. Archambeau, "Using subspace-based template attacks to compare and combine power and electromagnetic information leakages," *CHES 2008*, vol. 5154, E. Oswald and P. Rohatgi, Eds. Berlin, Germany: Springer-Verlag, 2008, pp. 411–425, ser. Lecture Notes in Computer Science.
- [33] SimonsVoss Technologies AG, "References," 2012. [Online]. Available: <http://www.simons-voss.com/References.1163.0.html?&L=1>
- [34] SimonsVoss Technologies AG, "Manual for WAVENET-FUNKNETZWERK 3065," 2011. [Online]. Available: [http://www.simons-voss.de/fileadmin/media/produkte/Handbuch\\_WaveNet\\_Fun\\_knetzwerk\\_3065\\_D.pdf](http://www.simons-voss.de/fileadmin/media/produkte/Handbuch_WaveNet_Fun_knetzwerk_3065_D.pdf)
- [35] Microchip Technology Inc., "PIC16F882/883/884/886/887 Data Sheet," 2009. [Online]. Available: <http://ww1.microchip.com/downloads/en/devicedoc/41291f.pdf>
- [36] NXP Semiconductors, "User manual UM10204—I<sup>2</sup>C-bus specification and user manual, rev. 5," 2012. [Online]. Available: [http://www.nxp.com/documents/user\\_manual/UM10204.pdf](http://www.nxp.com/documents/user_manual/UM10204.pdf)
- [37] Microchip Technology Inc., "PICkit 2 development programmer/debugger." [Online]. Available: [http://www.microchip.com/stellent/idcplg?IdcService=SS\\_GET\\_PAGE&storeId=1406&dDocName=en023805](http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&storeId=1406&dDocName=en023805)
- [38] Yubico, "Yubico reference customers: Department of Defense." [Online]. Available: <http://www.yubico.com/about/reference-customers/departement-defence/>
- [39] Yubico, "How YubiKeys are manufactured." [Online]. Available: [https://www.youtube.com/watch?v=s8\\_I1-ErZSQ](https://www.youtube.com/watch?v=s8_I1-ErZSQ)
- [40] Sunplus Innovation Technology Inc. [Online]. Available: <http://www.sunplusit.com>
- [41] Langer EMV-Technik, "LF1 near field probe set." [Online]. Available: <http://www.langer-emv.de/en/products/disturbance-emission/near-field-probes/lf-1/>
- [42] Langer EMV-Technik, "Preamplifier PA 303." [Online]. Available: <http://www.langer-emv.de/en/products/disturbance-emission/preamplifier/pa-203-303/devices-data/>

- [43] K. S. Shanmugam, *Digital & Analog Communication Systems*. Mumbai, India: Wiley-India, 2006.
- [44] J. W. Bos, D. A. Osvik, and D. Stefan, "Fast implementations of AES on various platforms," *IACR Cryptology ePrint Archive*, 2009, p. 501. [Online]. Available: <http://eprint.iacr.org/2009/501>
- [45] D. Oswald and C. Paar, "Improving side-channel analysis with optimal linear transforms," *Smart Card Research and Advanced Applications*, vol. 7771, S. Mangard, Ed. Berlin, Germany: Springer-Verlag, 2013, pp. 219–233, ser. Lecture Notes in Computer Science.
- [46] J. Balasch, B. Gierlichs, R. Verdult, L. Batina, and I. Verbauwhede, "Power analysis of Atmel CryptoMemory—Recovering keys from secure EEPROMs," *CT-RSA 2012*, vol. 7178, O. Dunkelmann, Ed. Berlin, Germany: Springer-Verlag, 2012, pp. 19–34, ser. Lecture Notes in Computer Science.
- [47] A. Moradi, A. Barengi, T. Kasper, and C. Paar, "On the vulnerability of FPGA bitstream encryption against power analysis attacks: Extracting keys from Xilinx Virtex-II FPGAs," in *Proc. ACM Conf. Comput. Commun. Security*, 2011, pp. 111–124.
- [48] A. Moradi, M. Kasper, and C. Paar, "Black-box side-channel attacks highlight the importance of countermeasures—An analysis of the Xilinx Virtex-4 and Virtex-5 bitstream encryption mechanism," *CT-RSA 2012*, vol. 7178, Berlin, Germany: Springer-Verlag, 2012, pp. 1–18, ser. Lecture Notes in Computer Science.
- [49] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," *CHES 2012*, vol. 7428, E. Prouff and P. Schaumont, Eds. Berlin, Germany: Springer-Verlag, 2012, pp. 23–40, ser. Lecture Notes in Computer Science.
- [50] A. Moradi, D. Oswald, C. Paar, and P. Swierczynski, "Side-channel attacks on the bitstream encryption mechanism of Altera Stratix II: Facilitating black-box analysis using software reverse-engineering," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, 2013, pp. 91–100.
- [51] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," in *Proc. 28th Eur. Solid-State Circuits Conf.*, 2002, pp. 403–406.
- [52] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," *CRYPTO 1999*, vol. 1666, M. Wiener, Ed. Berlin, Germany: Springer-Verlag, 1999, pp. 398–412, ser. Lecture Notes in Computer Science. [Online]. Available: [http://dx.doi.org/10.1007/3-540-48405-1\\_26](http://dx.doi.org/10.1007/3-540-48405-1_26)
- [53] J.-S. Coron and L. Goubin, "On Boolean and Arithmetic Masking against differential power analysis," *CHES 2000*, vol. 1965, C. K. Koç and C. Paar, Eds. Berlin, Germany: Springer-Verlag, 2000, pp. 231–237, ser. Lecture Notes in Computer Science.
- [54] C. Tarnovsky, "Deconstructing a 'secure' processor," Presentation at Black Hat USA, 2010. [Online]. Available: [http://www.blackhat.com/html/bh-dc-10/bh-dc-10-speaker\\_bios.html#Tarnovsky](http://www.blackhat.com/html/bh-dc-10/bh-dc-10-speaker_bios.html#Tarnovsky)
- [55] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlie, and J.-P. Seifert, "Simple photonic emission analysis of AES," *CHES 2012*, vol. 7428, E. Prouff and P. Schaumont, Eds. Berlin, Germany: Springer-Verlag, 2012, pp. 41–57, ser. Lecture Notes in Computer Science.

## ABOUT THE AUTHORS

**Daehyun Strobel** received the B.S. degree in applied computer science and the M.S. degree in IT security from Ruhr University Bochum, Bochum, Germany, in 2006 and 2009, respectively, where he is currently working toward the Ph.D. degree with the Chair for Embedded Security.

His research interests include practical side-channel attacks on embedded systems and side-channel reverse engineering of embedded software.



**David Oswald** received the Ph.D. degree in IT security from the Horst Görtz Institute for IT-Security at the Ruhr University Bochum, Bochum, Germany, in 2013.

He is currently working for the Chair for Embedded Security, Ruhr University Bochum, Bochum, Germany. His main field of research is the practical security analysis of embedded systems. His focus is on attack methods that exploit weaknesses in the physical implementation of mathematically secure cryptographic algorithms. Those techniques include both (passive) side-channel analysis and (active) fault injection. He is the cofounder of Kasper & Oswald GmbH, Bochum, Germany, offering innovative products and services for security engineering.



**Bastian Richter** received the B.S. degree in IT security from Ruhr University Bochum, Bochum, Germany, in 2013, where he is currently working toward the M.S. degree in IT security.

He is a student assistant under the Chair for Embedded Security. His research interests include security of embedded systems with focus on side-channel analysis and fault injection.



**Falk Schellenberg** received the B.S. and M.S. degrees in IT security from Ruhr University Bochum, Bochum, Germany, in 2010 and 2012, respectively, where he is currently working toward the Ph.D. degree in physical security for the Chair for Embedded Security, under the supervision of C. Paar.

His research interests include (semi)invasive attacks, especially (optical) fault injection techniques, as well as practical attacks on real-world devices.



**Christof Paar** (Fellow, IEEE) received the M.Sc. degree from the University of Siegen and the Ph.D. degree from the Institute for Experimental Mathematics at the University of Essen, Germany.

He holds the Chair for Embedded Security at Ruhr University Bochum, Bochum, Germany, and is an Affiliated Professor at the University of Massachusetts Amherst, Amherst, MA, USA. He cofounded, with C. Koc, the Workshop on Cryptographic Hardware and Embedded Systems (CHES) series. He has over 150 peer-reviewed publications and is coauthor of the textbook *Understanding Cryptography* (New York, NY, USA: Springer-Verlag, 2010). He is a cofounder of ECRYPT—Embedded Security, a leading consultancy firm in applied security that is now part of Bosch. His research interests include highly efficient software and hardware realizations of cryptography, physical security, security evaluation of real-world systems, and cryptanalytical hardware.



# Trustworthiness of Medical Devices and Body Area Networks

*This paper surveys the threat landscape of medical embedded devices and the merits and shortcomings of existing defenses.*

By MENG ZHANG, ANAND RAGHUNATHAN, *Fellow IEEE*, AND NIRAJ K. JHA, *Fellow IEEE*

**ABSTRACT** | Implantable and wearable medical devices (IWMDs) are commonly used for diagnosing, monitoring, and treating various medical conditions. A general trend in these medical devices is toward increased functional complexity, software programmability, and connectivity to body area networks (BANs). However, as IWMDs become more “intelligent,” they also become less trustworthy—less reliable and more prone to attacks. Various shortcomings—hardware failures, software errors, wireless attacks, malware and software exploits, and side-channel attacks—could undermine the trustworthiness of IWMDs and BANs. While these concerns have been recognized for some time, recent demonstrations of security attacks on commercial products, e.g., pacemakers and insulin pumps, have elevated medical device security from the realm of theoretical possibility to an immediate concern. The trustworthiness of IWMDs must be addressed aggressively and proactively due to the potential for catastrophic consequences. Conventional fault tolerance and information security solutions, e.g., redundancy and cryptography, that have been employed in general-purpose and embedded computing systems cannot be applied to many IWMDs due to their extreme size and power constraints and unique usage models. While several recent efforts address defense of IWMDs against specific security attacks, a holistic strategy that considers all concerns and types of threats is required. This paper discusses trustworthi-

ness concerns in IWMDs and BANs through a comprehensive identification and analysis of potential threats and, for each threat, provides a discussion of the merits and inadequacies of current solutions.

**KEYWORDS** | Body area networks (BANs); medical devices; personal healthcare systems; privacy; reliability; security

## I. INTRODUCTION

Recent years have witnessed an explosion of activity in the development and use of implantable and wearable medical devices (IWMDs) for a variety of diagnostic, monitoring, and therapeutic applications. Advances in electronics promise to revolutionize the capabilities of IWMDs, leading to new generations of devices with increased functionality, programmability, and connectivity to body area networks (BANs). In addition, IWMDs are increasingly being connected to personal computers (PCs) and smartphones, and to web- or cloud-based medical data repositories, to provide patients with complete personal healthcare systems (PHSs). These advances, however, are shadowed by concerns about the trustworthiness—reliability and security—of the software and hardware deployed in such systems.

Due to their increasing functional complexity, ensuring the reliability of IWMDs is more challenging than ever. As devices become increasingly smaller in size, but more complex in both software and hardware, their design, testing, and eventual regulatory approval are becoming much more expensive for medical device manufacturers, both in terms of time and cost. The number of devices that have recently been recalled due to software and hardware defects is increasing at an alarming rate [1]. At the same time, the increasing programmability and

Manuscript received May 14, 2013; accepted April 23, 2014. Date of publication June 6, 2014; date of current version July 18, 2014. This work was supported by the National Science Foundation (NSF) under Grants CNS-0914787 and CNS-1219570.

M. Zhang and N. K. Jha are with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: mengz@princeton.edu; jha@princeton.edu).

A. Raghunathan is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: raghunathan@purdue.edu).

Digital Object Identifier: 10.1109/JPROC.2014.2322103

0018-9219 © 2014 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See [http://www.ieee.org/publications\\_standards/publications/rights/index.html](http://www.ieee.org/publications_standards/publications/rights/index.html) for more information.

network connectivity of IWMDs open them up to the possibility of malicious attacks. Recent demonstrations of successful attacks on medical devices, such as implantable cardioverter defibrillators (ICDs) [2] and insulin pumps [3], [4], have already placed them squarely in the focus of attackers.

Any concerns regarding trustworthiness in medical devices must be addressed aggressively and proactively due to the potential for catastrophic consequences. Unfortunately, IWMDs come with extreme size and power constraints, making it infeasible to simply borrow reliability or security solutions from the general-purpose computing arena. Therefore, this is an area that demands the immediate attention of the information security and embedded systems research communities, medical device manufacturers, and regulatory bodies.

This paper aims to provide a comprehensive analysis and categorization of the threats posed to the trustworthiness of IWMDs, and promising approaches to guard against them. In the following sections, we first provide some background on medical devices and BANs, and then analyze their reliability and security requirements. We next examine various types of threats that compromise the trustworthiness of these systems, and discuss suitable countermeasures against them.

## II. PERSONAL HEALTHCARE SYSTEMS

This section provides relevant background on medical devices, and discusses how they are connected to BANs and various computing platforms to form PHSs.

### A. Medical Devices

We start with an overview of medical devices.

The U.S. Food and Drug Administration (FDA) provides several classification standards for medical devices. Each device approved by the FDA is classified into one of the following 16 categories based on the medical specialty: anesthesiology; cardiovascular; clinical chemistry and clinical toxicology; dental; ear, nose, and throat; gastroenterology and urology; general and plastic surgery; general hospital and personal use; hematology and pathology; immunology and microbiology; neurology; obstetric and gynaecologic; ophthalmic; orthopedic; physical medicine; and radiology [5].

Based on the potential risk for causing harm, the FDA provides another way of classifying medical devices [6]. Devices with a minimal risk, such as tongue depressors and handheld surgical instruments, belong to class I. Class I devices are subject to the least regulatory control: general controls. Medical devices with a higher risk, such as wheelchairs, surgical needles, and infusion pumps, are classified into class II. In addition to complying with general controls, class II devices are also subject to special controls, including special labeling, mandatory performance standards, postmarket surveillance, etc. Class III devices are more

invasive and pose a much more significant risk, against which neither general nor special control is sufficient to assure safety and effectiveness. Such a device requires premarket approval, in addition to the general controls. Examples of such devices include replacement heart valves, cardiac pacemakers, and neurostimulators.

In this paper, we limit ourselves to IWMDs of two kinds: sensors that monitor the patient's ECG, temperature, blood glucose and oxygen levels, etc., and actuators that deliver therapies, such as cardiac pacing and drug injection. Sensors and actuators are often combined into a closed-loop system. For example, a pacemaker implanted into the human body includes a sensor that performs pacing measurements, and also an actuator that provides rate-adaptive bradycardia pacing support by analyzing the sensed data.

An actuator is usually equipped with a programmer to change configurations or issue commands wirelessly. For certain devices, such as pacemakers, programmers are available only in clinics or hospitals, and the patient must visit a qualified healthcare provider for device tuning. For other devices, such as insulin pumps, patient programmers are available that allow patients to adjust the devices to meet their needs at any time.

### B. From IWMDs to BANs and PHSs

Fig. 1(a) and (b) presents some applications of IWMDs, including cardiac pacing, epileptic seizure detection, diabetes management through glucose monitoring and insulin delivery, etc. Fig. 1(c) presents a generic architecture for how IWMDs can be connected with each other via a BAN, as well as other computing platforms (mobile devices, PCs, and servers) to form a PHS. The system consists of four parts: medical sensors/actuators/programmers, a "hub" such as the patient's smartphone or a PC, a remote health server, and the doctor's smartphone or a PC.

The BAN facilitates communication among IWMDs and the hub. IWMDs may communicate with a patient's hub or with diagnostic equipment used by a healthcare provider, using short-distance communication technology, such as Medical Implant Communications Service [7], Bluetooth [8], ZigBee [9], or Ultrawideband Radios [10].

The hub is responsible for logging, compressing, and analyzing the raw health data recorded by the medical sensors. It serves as a bridge between the BAN and cellular service or an 802.11 wireless local area network (WLAN) to enable interaction between patients and remote healthcare providers. If any anomalies are detected in the readings from the sensors, the warnings and patient's location can be immediately transmitted to the healthcare provider.

Long-term data archival may be performed on a health server [11]–[13], which can be accessed by the patient or the healthcare provider. For example, a remote doctor can access the health server, query the patient's medical data, and provide a prescription or instructions, which may be displayed on the patient's hub.

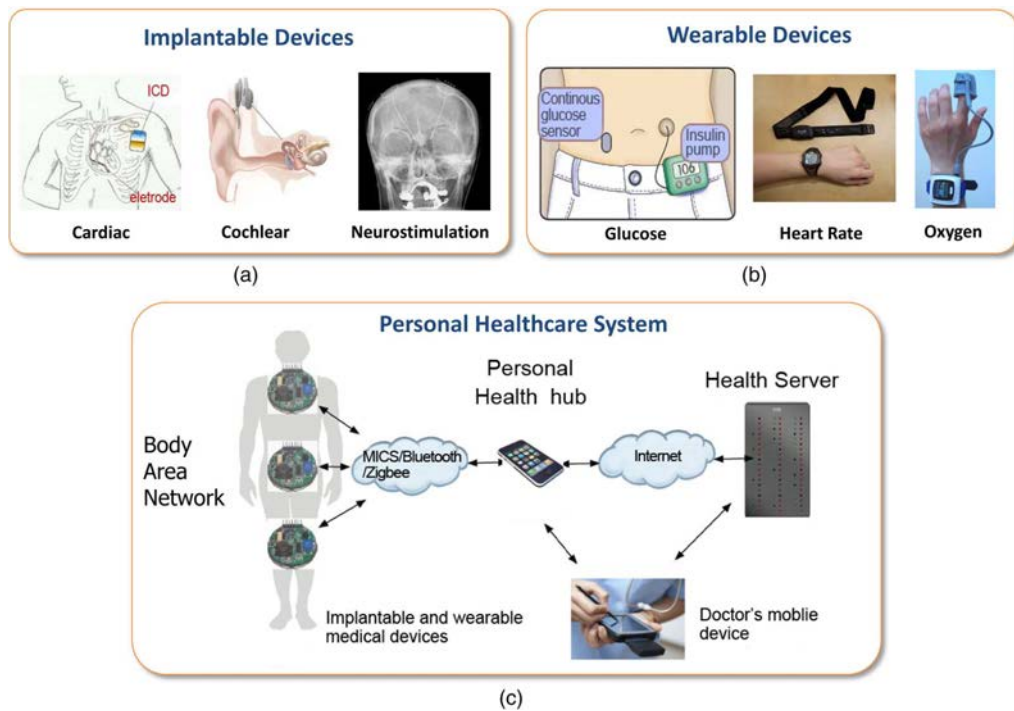


**Fig. 1.** (a) Implantable and (b) wearable medical devices (adopted from <http://www.wikipedia.org/>), and (c) integration of IWMDs into a PHS.

### C. PHS Trustworthiness Requirements

Unfortunately, any discussion on the potential of IWMDs and PHSs must be tempered by a sobering concern—trustworthiness. Studies suggest that security and safety are among patients' top concerns regarding IWMDs [14]. The sensitive nature of medical data and catastrophic consequences of safety failures or security breaches of PHSs make conventional reactive approaches unacceptable. Even when the consequences of a security attack are not life-threatening, the resulting negative perception and loss of trust may significantly slow down technology adoption. For example, the poor adoption of online health record services [11], [12] has been attributed in large part to security concerns [15].

Table 1 summarizes major PHS trustworthiness requirements.

1) *IWMD Reliability*: Obviously, devices that perform life-sustaining functions must be reliable. If the device only provides monitoring capability and treatment is decided based on further assessment, the result of malfunction may not necessarily be harmful. For example, the wearable fall detector [16] for the elderly detects the occurrence of an unintentional fall and provides the location of the victim. A malfunctioning detector may send false alarms when the patient is safe and sound. In this case, the malfunction is apparent and the detector can be easily replaced. However, even for noncritical devices, malfunction can result in catastrophic consequences in indirect

ways. For example, a visually impaired person aided by an artificial retina prosthesis [17] may suddenly lose vision while driving or crossing the street due to retina sensor failure.

2) *BAN Security*: Information leaked during wireless transmission can compromise BAN confidentiality and patient privacy. For example, the glucose monitoring and insulin delivery system discussed in [3] broadcasts the patient's diabetic condition when wirelessly transmitting

**Table 1** PHS Trustworthiness Requirements

| Requirements    | Description                                                                                                                                                                                 |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Reliability     | – IWMDs should function correctly even under extreme environmental conditions.                                                                                                              |
| Confidentiality | – Information transmitted within the BAN should be secured from access by unauthorized parties.<br>– Patient data stored on IWMD, health hub, or remote server should be kept confidential. |
| Integrity       | – Information transmitted within the BAN should be authentic and complete.<br>– Patient data stored on IWMD, health hub, or remote server should not be altered.                            |
| Availability    | – The BAN should be available even during jamming and denial-of-service attacks.<br>– Patient data stored on IWMD, health hub, or remote server should be readily retrievable.              |
| Privacy         | – Using a PHS or carrying a device should not disclose patient condition.                                                                                                                   |

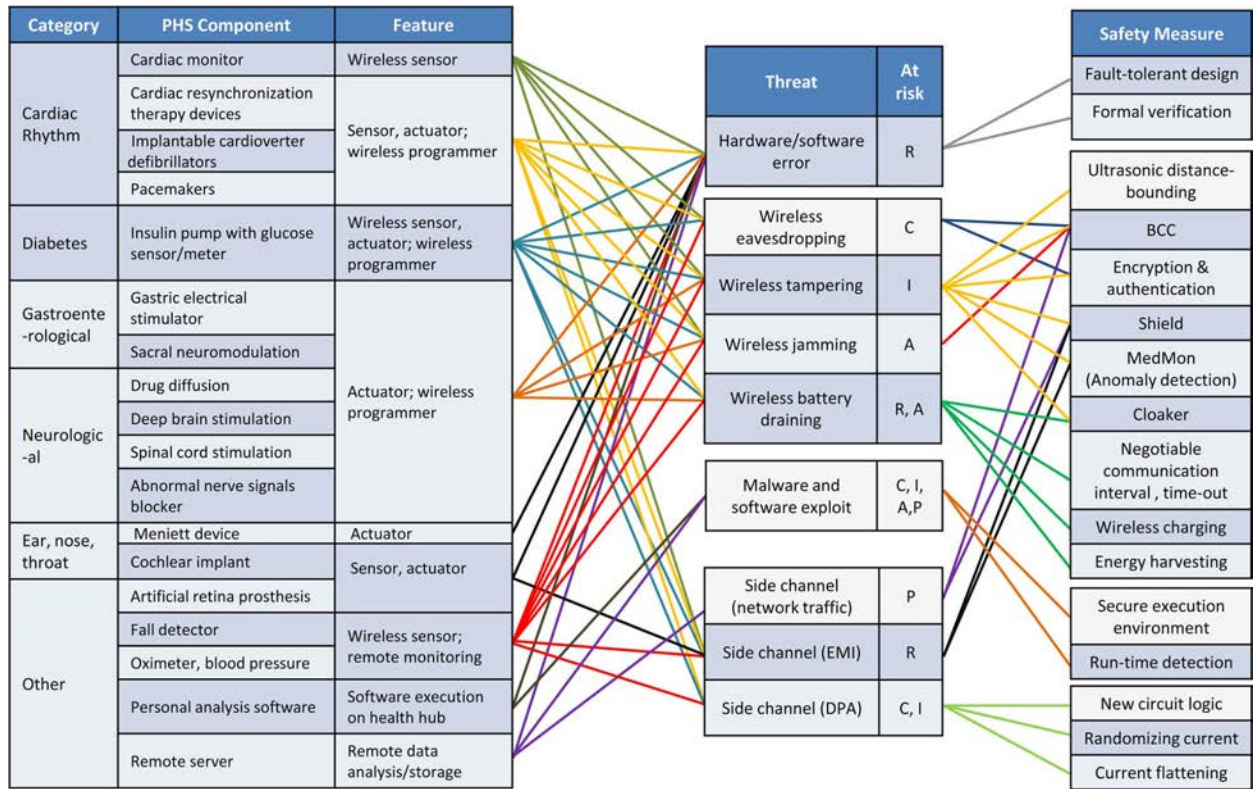


**Fig. 2. IWMDs, threats, and countermeasures.** Reliability, confidentiality, integrity, availability, and privacy are referred to as R, C, I, A, and P, respectively.

unencrypted data and commands. Even worse, an attacker can use the information (such as the device PIN) gained from eavesdropping on such communications to launch attacks and compromise BAN integrity. Depending on the device functionality, the consequence of an integrity compromise can be catastrophic, e.g., a cardiac pacemaker whose setting is changed by an attacker [2].

In addition to confidentiality and integrity requirements, the BAN and devices within it should be able to withstand attacks that are intended to cause unavailability, e.g., jamming attacks that flood the wireless channel, and denial-of-service attacks that drain the device battery [18].

3) *Data Security*: Just like BAN security, data security entails data confidentiality, integrity, and availability. Patient data should be kept safe when stored either on IWMDs, the hub, or the remote server. A compromised hub or a remote server can lead to a privacy breach. In addition, if treatment is decided based on the monitored data, corrupted data may lead to unsafe therapy, and missing/irretrievable data may impact the timeliness of treatment.

4) *Privacy*: In the context of this paper, privacy refers to hiding the patient condition from unauthorized parties. Privacy infringement can be a side effect of BAN/data

confidentiality compromise, but may also occur in more subtle ways. It is important for IWMD manufacturers and healthcare providers to strive to protect patient privacy. For example, while a wheelchair user may not expect much privacy, it is understandable if a cancer patient wearing a tumor monitor [19] prefers to keep his condition a secret. If the attacker has knowledge of the wireless communication protocol used, the transmissions from an IWMD easily reveal its presence, and so does a response to a communication request sent by an attacker.

Fig. 2 shows examples of potential threats that can place a PHS at risk. Column 1 lists IWMDs and other PHS components. Column 2 shows the threats they are prone to. Column 3 includes possible defenses against the particular threats. We discuss these topics in detail next.

### III. VULNERABILITY ASSESSMENT AND COUNTERMEASURES

In this section, we examine the various threats faced by IWMDs and other PHS components, all of which are capable of compromising a PHS and making its operation unsafe. These threats may arise from hardware/software errors or malicious attacks, which include radio attacks, side-channel attacks, malware attacks, and vulnerability

exploits. We also discuss countermeasures against the threats.

Each of the following sections addresses one type of threat. Suitable countermeasures as well as their merits and inadequacies are also discussed. Some of these solutions are widely used, or have even been included in standards, whereas others are based on recent research.

### A. Hardware Failures/Software Errors

In general, a hardware failure can be in electronic or nonelectronic components. Examples of nonelectronic hardware failure include contaminated syringes and broken wheelchairs. We focus on electronic hardware failure. Such a failure can be caused by undetected manufacturing defects, wear-and-tear faults due to electromigration, hot carrier degradation, dielectric breakdown, etc., as well as transient errors induced by a complex physical environment (e.g., due to noise, power disturbance, extreme temperature, vibration, electromagnetic interference, etc.). Studies have shown that electromagnetic interference may cause temporary or permanent malfunction in pacemakers and ICDs [20], [21].

Many IWMDs perform life-sustaining functions, such as cardiac pacing and defibrillation, and insulin delivery. They are also responsible for monitoring, recording, and storing private patient information, and making changes in response to doctors' orders. The critical nature of their functionality and the fact that they are in close contact with human organs leave little tolerance for hardware failure. A glitch on a cellphone may go unnoticed, whereas a glitch on a pacemaker or an ICD can be life-threatening. Besides cardiovascular and diabetic devices, neuromodulation devices that treat neurological conditions, such as Parkinson's disease and epilepsy [22], [23], are also examples that call for high standards for robustness and reliability. In a typical deep-brain stimulation system from Medtronic [24], several leads with electrodes are implanted in the brain and connected to a neurostimulator implanted near the clavicle. The amount of electrical stimulation can be noninvasively adjusted by a programmer. However, electromagnetic interference, if strong enough, could also change the parameter values of the neurostimulator, turn the neurostimulator off, or cause it to give the patient a jolt. A brain-machine interface (BMI) is expected to be at the center of the next generation of neuromodulation devices. It provides a direct communication pathway between the brain and an external device. BMI studies have shown promise for memory augmentation as well as perceptual and motor prostheses [25]. However, in addition to the technical challenges posed by interfacing of electronics with neurons, reliability concerns remain. Whether BMIs can reliably interact with the complex nervous system without being disrupted by environmental interferences needs further study.

Software reliability is just as important as hardware reliability. Many IWMDs are essentially embedded sys-

tems and have significant software content. Any safety and regulatory requirements for medical devices necessarily call for a rigorous software development process and skilled engineers in order to minimize software errors and protect public health. Unfortunately, designing bug-free software is difficult, especially in complex devices that might be used in unanticipated contexts. What is worse, software errors do not necessarily manifest themselves during the development and testing phase and may only result in errors after deployment.

More than a fourth of the recalls of defective medical devices during the first half of 2010 were likely caused by software defects [26]. Currently, there are no widely accepted techniques in use for the development or verification of software in medical devices [27]. The FDA is responsible for evaluating the risks of new devices and monitoring the safety and efficacy of those currently on the market. However, the FDA only assesses the development process of device software, not the integrity of the software itself. Unless a device that has already been surgically implanted repeatedly malfunctions or is recalled, the agency is unlikely to scrutinize the software operating on the device. Verification may often just depend on testing the device with crafted test cases, with little regard to the properties of the actual code. It has been argued that perhaps using open-source software is more secure and reliable for medical applications, as it enables continuous and broad peer review that identifies and eliminates software errors [26]. However, understandably, medical device manufacturers may be reluctant to adopt this approach.

*Solutions:* Next, we discuss two widely used techniques for enhancing hardware/software reliability.

1) *Fault-Tolerant Design:* Complex electronic circuits are used in life-critical healthcare systems, where reliability is of paramount importance. Though manufacturing-time test typically identifies a large number of circuit defects, exhaustive testing and attaining complete fault coverage may not be feasible. Through the concurrent detection, diagnosis, and correction of fault effects, fault-tolerant designs enable a system to continue operating properly in the event of faults in its components [28]. Fault tolerance can also be extended to cope with software errors caused by design inadequacies [29].

In general, fault tolerance requires some form of redundancy, either in time, hardware, or information. Hence, it incurs either performance degradation or hardware overhead. For example, triple modular redundancy (TMR) [30], a well-known fault tolerance scheme that employs three copies of a module and uses a majority voter to determine the final output, has more than three times the cost of the original circuit. Despite their high cost, fault-tolerant design techniques may be warranted in safety-critical medical devices.

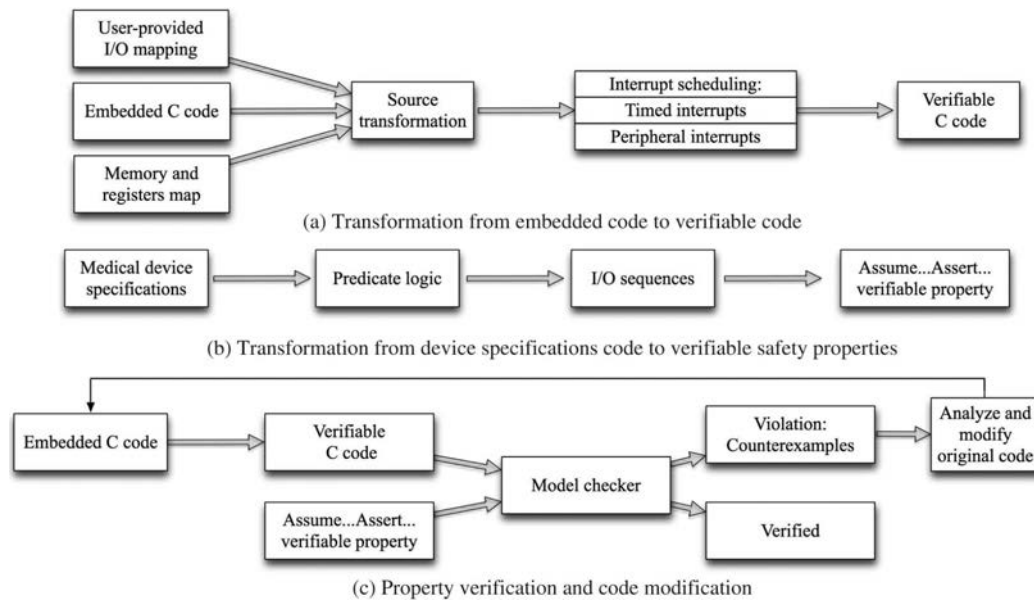


**Fig. 3. Methodology for verifying safety properties of medical device software [34].**

2) *Formal Verification*: Formal methods have been suggested as a means to design and develop reliable medical systems [27], [31], [32]. Formal methods are mathematical techniques for the specification, development, and verification of software and hardware systems [33]. The specifications used in formal methods are well-formed statements in a mathematical logic, and formal verification consists of rigorous deductions in that logic. Therefore, formal methods provide a means to symbolically examine the entire state space of a system and establish that a correctness or safety property is true for all possible inputs.

Formal verification may be used to ensure that the software running on medical devices is free of vulnerabilities, such as buffer overflows. However, this is far from sufficient to ensure that the medical device would operate in a trustworthy manner. Two key challenges must be addressed in order to truly leverage the power of formal verification in the context of medical devices [34]. First, current software verification tools target specifications written in high-level programming languages, and are not suitable for the highly platform-specific and low-level programs that are written for medical devices. These programs interact with hardware peripherals, such as medical sensors and actuators, in addition to timers, ADCs, UARTs, etc. In addition, they often adopt a highly interrupt-driven software architecture. It is necessary to verify the operation of these programs with sufficient semantics of the hardware platform that they execute on, while avoiding the state-space explosion that results from excessive detail in modeling the hardware. Second, properties need to be verified at the interfaces of the medical devices with the real world. In other words, rather than

merely verifying that a program is free of buffer overflows, it is equally, if not more so, important to verify whether any execution path in the pacemaker program leads to missing a cardiac pacing signal within a specified time window. A methodology to utilize formal verification for medical device software is shown in Fig. 3. The medical device software is first subject to a source-to-source transformation to address the aforementioned “semantic gap.” Properties based on the functional specification of the medical device are then expressed in terms of input/output (I/O) interactions of the medical device with its environment, and translated into equivalent properties or assertions that must be satisfied by the medical device software. The transformed code and verifiable assertions are fed to a model checker, which verifies the code against the assertions and reports whether the code has been verified or a violation has been found.

## B. Radio Attacks

A common IWMD design fallacy is relying on proprietary protocols for secrecy [35]. Since often no cryptographic protection is employed, wireless channels between devices and external controllers [e.g., the link between sensors and a smartphone in Fig. 1(c)], and between devices that communicate with each other, are highly prone to attacks.

A successful attack on an ICD is demonstrated in [2], which shows how the ICD design, which involves wireless communication with an external programmer, can be exploited by an attacker. By reverse-engineering the communication protocol, the attacker can launch radio attacks, with consequences ranging from disclosure of private data

to alteration of device settings. In the worst case, the attacker can maliciously reconfigure the ICD to harm a patient by inaction (failure to deliver treatment when necessary) or by delivering an electrical stimulus when the heart is beating normally.

Using a similar approach, the study in [3] implements a successful attack on a glucose monitoring and insulin delivery system, exploiting both the wireless channel between the device and external controller, and the wireless channel between devices. The attacker first eavesdrops on the wireless packets sent from a remote control to an insulin pump, and reverse-engineers the communication protocol. The same eavesdropping attack is performed on a glucose meter that sends the glucose-level data to the insulin pump. The attacker discovers the PINs associated with the remote control and glucose meter. By mimicking the remote control, the attacker can configure the insulin pump to disable or change the intended therapy, stop the insulin injection, or inject a much higher dose than necessary. By mimicking the glucose meter, the attacker can send bogus data to the insulin pump, causing the patient to incorrectly adjust insulin delivery.

Another such man-in-the-middle attack is demonstrated on a Bluetooth-enabled pulse oximeter system in [36]. With the assumption that the PIN used in standard Bluetooth pairing is known, the attack shows that these wearable devices can be made to communicate with an unauthenticated intermediary equipped with a Bluetooth-enabled laptop.

Finally, with the knowledge of the communication protocol, denial-of-service attacks that aim to drain the battery of an implantable medical device (IMD) may be launched through the wireless channel. If the device responds to each incoming communication request from attackers, its battery may simply die and need to be surgically replaced. In addition, an attacker could also generate a large amount of noise to jam normal communication if he simply knows the approximate frequency of transmission.

**Solutions:** Next, we discuss several methods to detect, defend against, or mediate radio attacks. They are classified into four categories: close-range communication, cryptography, external device, and battery-constraint mitigation.

1) *Close-Range Communication:* Limiting the communication range is a simple and intuitive way of limiting radio attacks. A radio-frequency identification (RFID)-based channel between medical devices and external controllers is often proposed in this context [37]. However, an attacker with a strong enough transmitter and a high-gain antenna can attack the wireless channel even if the channel is only for RFID-based communication. For an RFID channel, the attacker can access the IWMD from up to ten meters away [38], [39]. A better alternative may be near-field communication (NFC), an extension of RFID, which

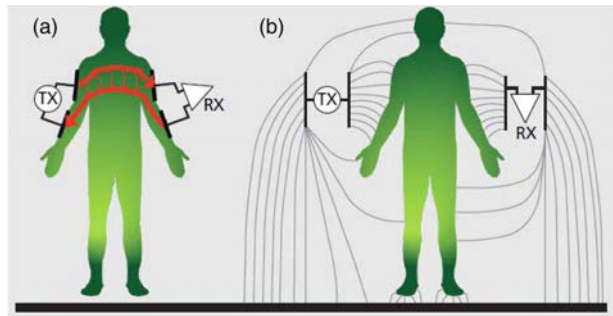


**Fig. 4. Two coupling mechanisms for body-coupled communication [42].**

is gaining increasing attention, especially due to its integration on mobile phones [40]. The typical working distance for NFC is up to 20 cm. However, there is no guarantee that an attacker with a high-gain antenna cannot read the signal from outside the intended range, e.g., from 1 m away [41].

Another technology that can help limit the communication range is body-coupled communication (BCC). In contrast to conventional wireless communication, BCC uses the human body as the transmission medium. The communication range is limited to the proximity of the human body. Fig. 4 illustrates two coupling mechanisms for BCC. In Fig. 4(a), electrodes are directly attached to the human body for transmitting electrical signals. In Fig. 4(b), the human body acts as a floating conductor, whose electric potential changes with the electric field generated by the transmitter, which is detected by the receiver [42]. Experimental results presented in [3] show a promising attenuation in signal strength measured from some distance when comparing the BCC channel signal to the air channel signal. However, these radios work at low frequencies (ranging from 10 kHz to 10 MHz) and can only achieve very low data rates.

In addition to communications that are designed to be inherently short range, measures can be taken to enforce close-range communication. An access control scheme based on ultrasonic distance bounding is introduced in [43]. In this scheme, an IWMD grants access to its resources to only those devices that are close enough. Shielding is another way of enforcing close-range communication. A metal shield that restrains wireless signals from traveling beyond it can effectively eliminate radio eavesdropping attackers at a distance. However, limiting the communication range is only effective against radio attacks launched from beyond a certain distance. It is quite possible that an attacker can approach within a small distance of the patient and even make physical contact without raising suspicion (e.g., in a crowded subway station). Therefore, close-range communication schemes cannot defend against all close-range attacks.

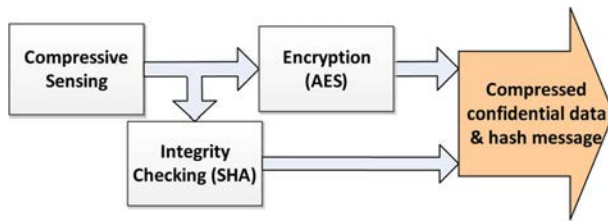


**Fig. 5. Encompression based on compressive sensing [45].**

2) *Cryptography*: Cryptography is the best approach for securing the wireless communication channel and preventing unauthorized access. The high energy and implementation costs of asymmetric cryptography preclude its use for encrypting medical data in IWMDs, which leaves symmetric encryption as the only practical option. However, the use of symmetric ciphers may still greatly increase the energy consumption and thus shorten device battery time. To mitigate this problem, compression techniques can be used before encryption to reduce this added workload as well as transmission cost. Compressive sensing [44] is particularly well suited since the compression can be realized with a very low computational and energy footprint. An evaluation of encompression (compressive sensing + encryption + integrity checking) [45] shows that an energy reduction of up to 78% can be achieved using encompression versus traditional encryption and integrity checking, with a reasonable compression ratio of 6–10 $\times$ . An overview of the encompression scheme is shown in Fig. 5. The advanced encryption standard (AES) [46] is used as the symmetric cipher and the secure hash algorithm (SHA) [47] is used for integrity checking. Note that the hash algorithm must be applied on the original data (the plaintext), because then imposters cannot generate encrypted data whose plaintext matches the hash output, without knowing the AES secret key. If it is applied on the encrypted data (the ciphertext), imposters can hash and send random spurious data if they have knowledge of the hash function used.

Cryptographic methods are even more attractive when the secret keys shared by IWMDs and the hub can be renewed periodically, as in 802.11 WiFi [48]. Fixed preconfigured keys, which are discussed in Section III-D, are prone to attacks. Furthermore, the secret keys should be updated automatically, since many users, such as the elderly, are unable or unwilling to configure keys of sufficient strength, or update them frequently. Ideally, shared keys should be generated with high agreement (low mismatch rate between the two communication parties), high randomness, at a fast rate, and with a minimum computational/energy overhead. Unfortunately, as always, tradeoffs must be made among these conflicting goals. Refreshing at a low rate (e.g., 1 b/s) can lower disagreement, computational/energy overhead, and improve ran-

domness, and may just suffice for low-data-rate IWMDs. Works described in [49]–[51] focus on extracting shared secret keys over unsecured wireless channel using the directional symmetry of wireless links. The most commonly used metric in these works is the received signal strength indicator (RSSI), a measure of signal power in logarithmic units. However, generation of randomness relies on relative movement between sensors and the hub, or the environment being dynamic [51]. In the case where both IWMDs and the hub are mounted on the body at fixed positions, the measured RSSI traces may not provide enough secrecy. In addition, rapid fluctuations in signal strength due to blockage by clothes or sudden movement may cause asymmetry in signals received at the two ends of the communication, which may result in key disagreement and require reconciliation.

Authentication schemes using properties of human bodies as alternatives to input of passwords have been proposed. IMDGardian [52] introduces an alternative cryptographic scheme for implantable cardiac devices that utilizes the patient's electrocardiography signals for key extraction. The identification system presented in [53] successfully recognizes people with an accuracy of 90% by measuring their bioimpedance to alternating current of different frequencies.

Unfortunately, conventional cryptographic methods are not directly applicable to IWMDs, whose unique usage models may require key distribution to legitimate parties outside the BAN. For example, encryption prevents medical professionals from accessing the patient's health data in emergency situations. As a possible solution, a universal key may be preloaded in devices of the same model that the ambulance staff can request from the manufacturer or patient's doctor in emergencies. However, this scheme is inherently unsafe as attackers can discover the secret key of a particular model through side-channel attacks or by hacking into the doctor's computer. Another straightforward key-distribution solution is to ask patients to carry cards or bracelets imprinted with the secret keys of their devices. To prevent the imprints from being lost or damaged, the keys could be printed into the patient's skin using ultraviolet-ink micropigmentation [54]. These "tattoos" only become visible under ultraviolet light, which is how the ambulance staff can find the keys and access the devices. To some extent, this approach is secure against close-range attacks, since although the attacker may be in close proximity, it is unlikely that the attacker can lift up the patient's sleeves while shining ultraviolet light without raising suspicion.

3) *External Device*: To preserve IMD battery power, verification of incoming requests can be offloaded to a trusted external device, which, unlike IMDs, can be easily recharged. A wearable device, called Communication Cloaker, is described in [55]. The Cloaker mediates communications between the IMD and preauthorized parties

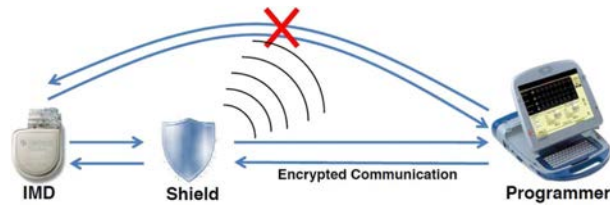


**Fig. 6.** External device that relays communications between the IMD and the programmer [56].

and causes the IMD to ignore incoming communications from all unauthorized programmers. If the Cloaker is missing or broken, the IMD accepts and responds to all incoming communications. Therefore, in emergency situations, the medical staff can remove the Cloaker in order to access the IMD. Since the burden of computation is off-loaded to the external device, this approach can protect the IMD against battery-draining attacks.

Another external device, a personal base station called the “shield,” is described in [56]. It is illustrated in Fig. 6. The shield works as a relay between the IMD and the external programmer. It is designed to receive and jam the IMD messages at the same time, so that others cannot decode them. It then encrypts the IMD message and sends it to the legitimate programmer. The shield also protects the IMD from unauthorized incoming commands by jamming all the messages sent directly to the IMD. All commands must be encrypted and sent to the shield first, which then relays legitimate commands to the IMD. Therefore, the shield does not require any change in commercial IMDs, but requires changes in all programmers. Since the messages from the IMD are jammed and the communication between the programmer and the shield is encrypted, the confidentiality of IMD messages is protected. However, when the shield sends programmer’s commands to the IMD, confidentiality is not warranted.

A medical security monitor (MedMon), proposed in [57], snoops on all the radio-frequency wireless communications to/from medical devices and uses anomaly detection to identify potentially malicious transactions. Anomalies are detected through physical characteristics of the transmitted signal, such as the RSSI, the time of arrival (TOA), the differential time of arrival (DTOA), and the angle of arrival (AOA), or behavioral characteristics embedded in the underlying information. Upon detection of a potential malicious transaction, MedMon takes appropriate response actions, which could range from passive (notifying the user) to active (jamming the packets so that they do not reach the medical device). Table 2 shows examples of security policies that could be used. These policies may be different for different IWMDs. The parameters mentioned in the table can be set to predefined values or else tailored to the patient’s condition and environment. Values of parameters associated with physical

**Table 2** Examples of Security Policies [57]

| Type       | Anomaly                                                                         | Response           |
|------------|---------------------------------------------------------------------------------|--------------------|
| Physical   | RSSI is greater than $A_h$ or smaller than $A_l$                                | Raise warning, jam |
|            | TOA does not fall in any of the time ranges $(t_l(m), t_h(m))$                  | Raise warning, jam |
|            | DTOA is greater than $\Delta t_h$ or smaller than $\Delta t_l$                  | Raise warning, jam |
|            | AOA is greater than $\alpha_h$ or smaller than $\alpha_l$                       | Raise warning, jam |
| Behavioral | Data value is greater than $D_h$ or smaller than $D_l$                          | Raise warning      |
|            | Rate of change of data value is greater than $r_{th}$ or smaller than $-r_{th}$ | Raise warning      |
|            | Injection dose is larger than $D_{th}$                                          | Raise warning, jam |
|            | Repeated $n$ injections in the past $\Delta T$                                  | Raise warning, jam |
|            | Total injected dose greater than $V_{th}$ in the past $\Delta T$                | Raise warning, jam |

anomalies, e.g.,  $A_h$  and  $\Delta t_h$ , can be generated automatically at the end of a setup period. Values of parameters associated with behavioral anomalies, e.g.,  $D_{th}$  and  $r_{th}$ , can be selected based on the advice of the doctor. By acting like a firewall, the monitor protects the BAN against integrity attacks, which are arguably the most dangerous type of attacks. It does not protect BAN confidentiality and patient privacy against eavesdropping, nor does it protect BAN availability against jamming. However, it does have the ability to provide protection for IWMD availability against battery-draining attacks, as transmissions that are too frequent may be seen as malicious and jammed. A key benefit of MedMon is that it is applicable to both existing IWMDs and programmers with no hardware or software modifications needed for them. Consequently, it leads to zero power overheads on these devices.

4) *Battery-Constraint Mitigation*: Compared to wearable devices whose battery can be readily recharged or replaced, battery-draining attacks pose much greater threat to IMDs, such as EEG implants and pacemakers, since replacing the battery usually implies surgery. Zero-power defenses (security at no cost to the battery) have been proposed for ICDs, in which the induced RF energy is harvested for notification, authentication, and key exchange [2]. In addition, efforts are being undertaken to design BAN protocols to mitigate this problem. For example, the IEEE 802.15.6 BAN standard allows a node and a hub to negotiate their communication intervals by encoding them in authenticated messages. The node thus will not wake up to receive any messages outside the negotiated time intervals [58].

Relaxing the battery constraints would be the best defense against battery-draining attacks. One solution is to make the implant wirelessly rechargeable [59]. Another is to harness kinetic energy from the human body [60].

However, as intriguing as these new techniques may sound, they are still in the research phase and must go through rigorous testing and examination to ensure trustworthiness before commercial use.

### C. Malware and Vulnerability Exploits

Various forms of malware, including viruses, worms, Trojans, keyloggers, botnets, and rootkits, have emerged and keep evolving and adapting to new platforms. Smartphone platforms, such as Android and iOS, have been breached by mobile malware [61], [62]. With the increasing flexibility and connectivity of PHS platforms, it is just a matter of time before the first appearance of malware that targets PHS platforms. For example, Intel Health Guide [13] is a chronic care product that delivers personalized health monitoring at home. Patients can use the system to measure their own vital signs and, through the Internet, upload the results to a remote server, where healthcare professionals can assess the patient's health condition. A virus that infects such a system can delete or forge health data.

Furthermore, since software is inherently complex, abstract, and intangible, software vulnerabilities are inevitable and difficult to detect. Software vulnerabilities differ from software errors in that they are not logical errors, but unsafe code segments that can be exploited by attackers. For example, use of especially crafted inputs, which trigger buffer overflows and redirect the program to execute malicious code, is called a buffer overflow attack. The corrupted memory could originally be holding an address to an instruction, which the program should be redirected to. After corruption of the address, the program may be redirected to a false address and start executing random code. With some knowledge of system software, attackers can exploit the buffer overflow vulnerabilities as well as other software vulnerabilities to steal private information, tamper with medical data, and even change device settings.

While BANs are subject to unique threat models and attacks, as described in previous sections, software attacks will continue to be a commonly utilized approach for compromising their security, due to the relative ease and low cost of launching such attacks. In this context, the "weakest link" of a BAN, i.e., the component that exposes the largest attack surface and is the most accessible to software attacks, is the health hub, which executes the medical applications (for logging of health data, display of data to the user, and communication with remote medical professionals and health information services). As reflected by the rapid proliferation of "application" marketplaces for mobile devices, users are likely to use their smartphones to execute untrusted and potentially vulnerable applications as well. In the extreme case, the operating system (OS) on the hub may itself be compromised, making it trivial to subvert the medical applications that execute under its full control. Thus, it becomes essential to provide a secure execution environment for the medical

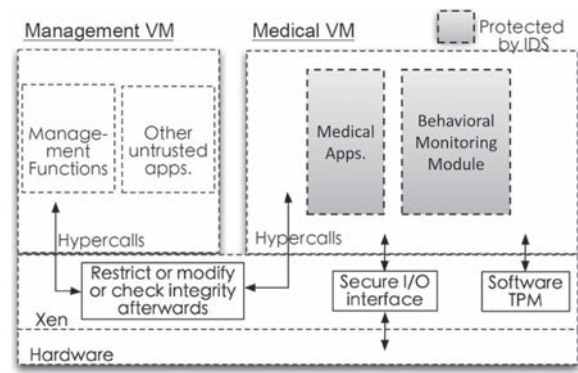


Fig. 7. Secure execution environment for medical applications.

applications in the face of other untrusted applications and also an untrusted OS.

*Solutions:* Two sets of techniques, secure execution environment and runtime monitoring, can be used to defend medical software/data against malware, vulnerability attacks, and malicious OS.

1) *Secure Execution Environment:* While it may not be feasible to secure all applications from a compromised OS, it is possible to achieve a secure execution environment that provides isolation for selected, security-critical applications. The isolation may be based on physical separation (e.g., IBM's secure coprocessor [63]) or logical separation, in which both the sensitive and untrusted codes are run on the same processor, but are isolated either using an additional layer of software, or through additional hardware support, such as ARM TrustZone [64].

A secure execution environment based on logical separation for medical applications is illustrated in Fig. 7. It is based on two key technologies: secure virtualization and trusted computing. For PHSs, virtualization is a promising technology that can be utilized to enhance security by providing isolated execution environments for different applications that require different levels of security. The medical applications are the most security-critical components. As shown in Fig. 7, they can be protected in a separate virtual machine (VM), which we refer to as the medical VM. The medical VM is a restricted environment in which only medical applications and the supporting software libraries are executed, isolated from the other applications running on the system. Trusted computing [65] is a set of standards that is widely gaining popularity in general-purpose computing systems. Trusted computing requires a "root of trust" in the system for tamper-proof storage and attestation, which is typically realized by adding a separate tamper-proof hardware component called the trusted platform module (TPM) to the system. In size-constrained and resource-constrained platforms, such as

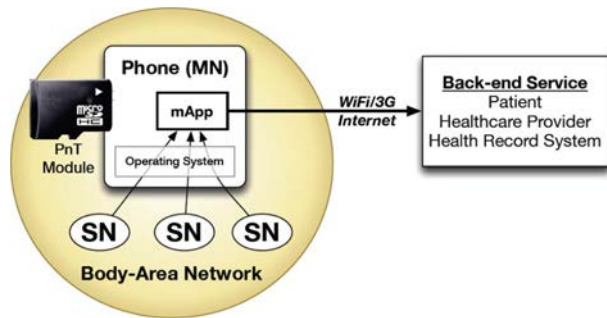


**Fig. 8.** Data separation by a plug-in smart card. The smartphone OS only has an encrypted view of medical data [67].

smartphones, it is currently not common to see hardware TPMs. In such cases, the use of a software TPM based on software emulation of TPM functions within an isolated execution environment has been demonstrated [66].

In addition to logical separation, the goal of data confidentiality and integrity can also be achieved by physically separated and secured data storage on the hub (smartphone). For example, Plug-n-Trust [67] is a plug-in smart card that provides a trusted computing environment and keeps medical data safe. The principle of Plug-n-Trust is as follows. Assuming the data sent by the medical sensors are encrypted, they remain encrypted while stored on the hub, and are only decrypted within the smart card. Application programming interfaces (APIs) are provided by the card to allow data modification by medical applications. This model is illustrated in Fig. 8.

A more aggressive approach is to completely separate health-related applications from untrusted applications/OS by making the hub an independent device. One such wrist-worn device, called Amulet (Fig. 9), is proposed in [68]. Amulet is dedicated to communications with IWMDs. It occasionally communicates with the smartphone in order to connect with health servers. It can also authenticate its wearer and determine which set of sensors

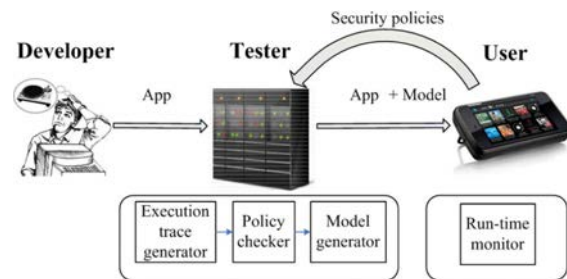


**Fig. 10.** Software runtime monitor. Applications are first tested in VMs and monitored when actually used.

are on the body by using techniques introduced in [69] and [70]. In addition to physical separation from potential software attacks, another strong argument in favor of a dedicated hub is that IWMDs must be able to operate continuously and securely without relying on smartphones or other nonwearable personal computing devices, which can easily be lost, stolen, or run out of power.

2) *Runtime Monitoring*: Isolating medical applications from other software does not protect against vulnerabilities in the medical applications themselves, which are commonly introduced into software as artifacts of the software development process. Intrusion detection techniques based on dynamic binary instrumentation have been extensively investigated [71]. As shown in Fig. 10, the application is first tested by running against a large input set (manually crafted or automatically generated) in a virtualized environment. If it passes the test, its behavioral models are generated, which can be seen as a database of good behaviors. The user may also request the publisher to test the application against user-defined policies that are of most interest to him/her. Runtime monitoring at the user end restricts the application's behavior to within the database of good behaviors. Any deviation is detected as an anomaly. As much of the workload is shifted from the user

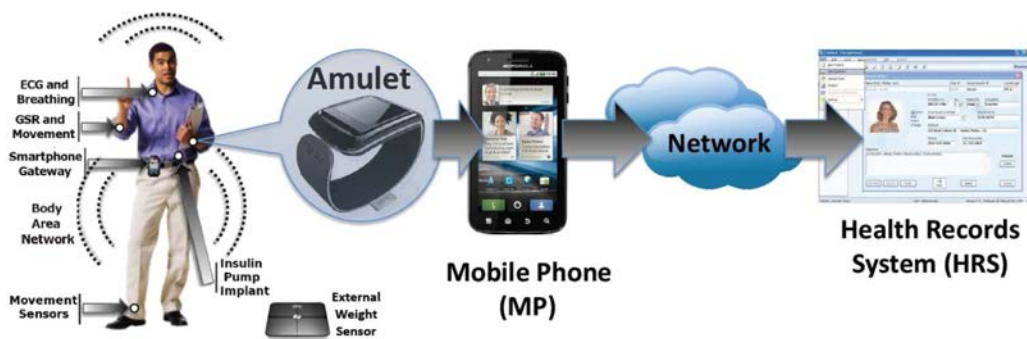


**Fig. 9.** Dedicated wrist-worn device as the hub [68].

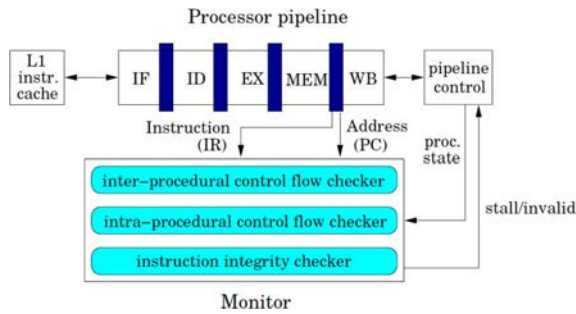


**Fig. 11. Hardware-assisted runtime monitor that monitors cycle-by-cycle trace of the executing instructions and their program addresses [72].**

to the testing end, the performance penalty is greatly reduced compared to rigorous runtime checking.

Even though the monitoring work is minimized, the delay overhead for such fine-grained monitoring can still be prohibitive, especially for applications with intensive user interactions. To overcome this problem, a hardware-assisted runtime monitor has been proposed for secure embedded processing [72]. Fig. 11 shows the conceptual block diagram of such a hardware monitor. The embedded processor is depicted as an in-order five-stage pipeline. It is augmented with a hardware monitor that observes the processor's dynamic execution trace, checks whether the execution trace falls within the allowed program behavior, and flags any deviations from the expected behavior to trigger appropriate response mechanisms. Program behaviors can be represented at different levels of granularity, namely, interprocedural control flow, intraprocedural control flow, and instruction stream integrity. When the monitor detects a violation of permissible program behavior, it asserts the invalid signal. In rare cases when the monitor is unable to keep pace with processor execution, it asserts the stall signal.

#### D. Side-Channel Attacks

Side-channel attacks exploit information leaked through physical channels, such as power consumption, execution time, electromagnetic emission, etc. [73]–[75]. They can be used against medical devices and PHSs for privacy invasion, as discussed in Section II-C4. For example, the Intel Health Guide system [13] is equipped with integrated cameras, allowing online health sessions and video consultations through the Internet. However, the network traffic flow may leak patients' private information. The schedule of health sessions and video calls, for example, could be deduced from monitoring the network traffic flow. One could also infer a change in the patient's health condition, if the lengths and frequencies of health sessions and video calls suddenly increase.

A more dangerous type of side-channel attack exploits electromagnetic interference (EMI), which can affect the

circuit by inducing voltage on conductors. Analog sensors in IWMDs are particularly susceptible. It has been shown that EMI can inhibit pacing and induce defibrillation shocks on implantable cardiac devices at a close distance [76].

Another form of side-channel attack is the differential power analysis (DPA) attack. A DPA attack can extract secret keys from extremely noisy signals and is very difficult to guard against. It employs statistical analysis of measured power consumption traces, which are correlated with the data handled by the physical device [73]. Although no known DPA attacks on any medical device have been reported, it is not hard to construct a scenario where DPA breaks cryptographic protection on IWMDs. Suppose a heart-rate monitor uses a symmetric block cipher (such as AES) with a built-in secret key to encrypt the measured heart rates before sending them to the hub. If an attacker gains access to the heart-rate monitor, the secret key can easily become a vulnerable target of DPA and extracted by feeding it with various data, measuring the corresponding current consumption, and analyzing the difference in measured current traces. Successful recovery of the secret key would then compromise confidentiality. Even worse, if a common default key is used for all shipped units of the same model, the attacker could publicize the revealed secret key and thus make the cryptographic protection ineffectual.

*Solutions:* Next, we introduce some of the proposed countermeasures against EMI and DPA attacks. Other types of side-channel attacks (e.g., cache attacks) and their countermeasures are omitted because we believe they are less applicable to the PHS model.

1) *Countermeasures Against EMI:* Shielding and filtering are commonly used defenses against EMI. In addition, cardiac defense mechanisms may take advantage of the physical proximity to the human body and detect suspicious sensor inputs by checking whether pacing pulses are consistent with the refractory period of cardiac tissue [76]. This method falls under the category of anomaly detection.

2) *Countermeasures Against DPA:* Software solutions against DPA, such as key masking [77], which attempts to randomize the secret key prior to each execution of the scalar multiplication under analysis, incur too much energy overhead. Assists from hardware design are usually proposed.

As the reason for the vulnerability of classical CMOS logic circuits to DPA attacks lies in the imbalance of charging and discharging behavior between 0-to-1 and 1-to-0 transitions, novel logic styles with data-independent power consumption have been proposed as circuit-level solutions to reduce the dependence of power dissipation on input patterns [78], [79].

Other system-level countermeasures either try to suppress the differential signal used in the DPA attacks or randomize the power profile. For example, in [80], a current flattening circuit is introduced. In [81], an additional circuit serving as a bandpass filter is added to the cryptosystem to suppress information leakage through the current supply pin. In [82], an internally generated random mask based on ring oscillators is used to dynamically change the power consumption. In [83], a dynamic voltage and frequency switching approach is adopted in which both the voltage and the clock frequency can be dynamically selected by a processor. Random delays are inserted in the datapath in [84] and FinFET back-gate biasing is introduced in [85] to randomize the power profile.

Unfortunately, in most of the aforementioned methods, DPA resistance still comes at the expense of large area and power overheads, which are not compatible with resource-constrained IWMDs. Low-cost DPA-resistant design is still an open problem.

#### IV. CONCLUSION

A general trend in IWMDs is toward increased functional complexity, software programmability, and wireless network connectivity. An undesirable, yet inevitable, side effect of these trends is that IWMDs and BANs are increasingly vulnerable to security attacks. Trustworthiness concerns may become a hindrance to further commercialization of IWMDs and BANs. We analyzed various aspects of threats faced by them and discussed suitable solutions for each threat. Given the critical functions IWMDs perform, these issues should be addressed aggressively and proactively by the manufacturers before market deployment. ■

#### Acknowledgment

The authors would like to thank Prof. S. Sur-Kolay for her valuable feedback on the paper.

#### REFERENCES

- [1] Stericycle Expert Solutions, *FDA Recall Trends for Food, Drugs, Devices*, 2011. [Online]. Available: <http://www.expertrecall.com>
- [2] D. Halperin et al., "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *Proc. IEEE Symp. Security Privacy*, May 2008, pp. 129–142.
- [3] C. Li, A. Raghunathan, and N. K. Jha, "Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system," in *Proc. IEEE Int. Conf. e-Health Netw. Appl. Services*, Jun. 2011.
- [4] J. Radcliffe, "Hacking medical devices for fun and insulin: Breaking the human SCADA system," in *Proc. Black Hat Technical Security Conf.*, Jul./Aug. 2011.
- [5] U.S. Food and Drug Administration, *Device Classification*, 2009. [Online]. Available: <http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/Overview/ClassifyYourDevice>
- [6] U.S. Food and Drug Administration, *General and Special Controls*, 2009. [Online]. Available: <http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/Overview/GeneralandSpecialControls>
- [7] H. S. Savci, A. Sula, Z. Wang, N. S. Dogan, and E. Arvas, "MICS transceivers: Regulatory standards and applications," in *Proc. IEEE SoutheastCon*, Apr. 2005, pp. 179–182.
- [8] Bluetooth. [Online]. Available: <http://www.bluetooth.org>
- [9] Zigbee alliance. [Online]. Available: <http://www.zigbee.org>
- [10] D. Porcino and W. Hirt, "Ultra-wideband radio technology: Potential and challenges ahead," *IEEE Commun. Mag.*, vol. 41, no. 7, pp. 66–74, Jul. 2003.
- [11] Google Health Service. [Online]. Available: <http://www.google.com/health>
- [12] Microsoft HealthVault. [Online]. Available: <http://www.healthvault.com>
- [13] Intel Health Guide, 2011. [Online]. Available: <http://www.careinnovations.com/products/healthguide>
- [14] T. Denning et al., "Human values and security for wireless implantable medical devices," in *Proc. Int. Conf. Human Factors Comput. Syst.*, Apr. 2010.
- [15] M. Kolbasuk McGee, "5 reasons why Google Health failed," *InformationWeek*, Jun. 29, 2011. [Online]. Available: <http://www.informationweek.com/news/healthcare/EMR/231000697>
- [16] J. Chen, K. Kwong, D. Chang, J. Luk, and R. Bajcsy, "Wearable sensors for reliable fall detection," in *Proc. IEEE Int. Conf. Eng. Med. Biol. Soc.*, Jan. 2005, pp. 3551–3554.
- [17] L. Schwiebert et al., "A biomedical smart sensor for the visually impaired," in *Proc. IEEE Sensors*, 2002, vol. 1, pp. 693–698.
- [18] T. Martin, M. Hsiao, D. Ha, and J. Krishnaswami, "Denial-of-service attacks on battery-powered mobile computers," in *Proc. IEEE Conf. Perv. Comput. Commun.*, Mar. 2004, pp. 309–318.
- [19] K. D. Daniel et al., "Implantable diagnostic device for cancer monitoring," *Biosens. Bioelectron.*, vol. 24, no. 11, pp. 3252–3257, Jul. 2009.
- [20] D. L. Hayes et al., "Interference with cardiac pacemakers by cellular telephones," *New England J. Med.*, vol. 336, no. 21, pp. 1473–1479, May 1997.
- [21] C. Jilek et al., "Safety of implantable pacemakers and cardioverter defibrillators in the magnetic field of a novel remote magnetic navigation system," *J. Cardiovasc. Electrophysiol.*, vol. 21, no. 10, pp. 1136–1141, Oct. 2010.
- [22] J. M. Bronstein et al., "Deep brain stimulation for Parkinson disease: An expert consensus and review of key issues," *Arch. Neurol.*, vol. 68, no. 2, pp. 165–171, Feb. 2011.
- [23] W. H. Theodore and R. S. Fisher, "Brain stimulation for epilepsy," *Lancet Neurol.*, vol. 3, no. 2, pp. 111–118, Feb. 2004.
- [24] Medtronic, 2011. [Online]. Available: <http://www.medtronic.eu/our-therapies>
- [25] A. Csavoy, G. Molnar, and T. Denison, "Creating support circuits for the nervous system: Considerations for brain-machine interfacing," in *Proc. Int. Symp. VLSI Circuits*, Jun. 2009, pp. 4–7.
- [26] K. Sandler, L. Ohrstrom, L. Moy, and R. McVay, *Killed by Code: Software Transparency in Implantable Medical Devices*, 2010. [Online]. Available: <http://www.softwarefreedom.org>
- [27] D. Arney, R. Jetley, P. Jones, I. Lee, and O. Sokolsky, "Formal methods based development of a PCA infusion pump reference model: Generic infusion pump (GIP) project," in *Proc. Joint Workshop High Confidence Med. Devices Softw. Syst. Med. Device Plug-and-Play Interoper.*, Jun. 2007, pp. 23–33.
- [28] W. R. Moore, "A review of fault-tolerant techniques for the enhancement of integrated circuit yield," *Proc. IEEE*, vol. 74, no. 5, pp. 684–698, May 1986.
- [29] B. Randell, "System structure for software fault tolerance," *SIGPLAN Not.*, vol. 10, pp. 437–449, Apr. 1975.
- [30] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM J. Res. Dev.*, vol. 6, pp. 200–209, Apr. 1962.
- [31] R. Jetley, S. P. Iyer, P. L. Jones, and W. Spees, "A formal approach to pre-market review for medical device software," in *Proc. Int. Conf. Comput. Softw. Appl.*, Sep. 2006, pp. 169–177.
- [32] L. Cordeiro, B. Fischer, H. Chen, and J. Marques-Silva, "Semiformal verification of embedded software in medical devices considering stringent hardware constraints," in *Proc. IEEE Int. Conf. Embedded Softw. Syst.*, May 2009, pp. 396–403.
- [33] C. M. Holloway, "Why engineers should consider formal methods," in *Proc. IEEE Digit. Avionics Syst. Conf.*, Oct. 1997, pp. 16–22.
- [34] C. Li, A. Raghunathan, and N. K. Jha, "Improving the trustworthiness of medical device software with formal verification methods," *IEEE Embedded Syst. Lett.*, vol. 5, no. 3, pp. 50–53, Sep. 2013.
- [35] W. Burleson, S. S. Clark, B. Ransford, and K. Fu, "Design challenges for secure implantable medical devices," in *Proc. Design Autom. Conf.*, 2012, pp. 12–17.
- [36] V. Pournaghshband, M. Sarrafzadeh, and P. Reiher, "Securing legacy mobile medical devices," in *Proc. Int. Conf. Wireless Mobile Commun. Healthcare*, 2012.
- [37] C. Israel and S. Barold, "Pacemaker systems as implantable cardiac rhythm monitors,"

- Amer. J. Cardiol., vol. 88, no. 4, pp. 442–445, Aug. 2001.
- [38] K. Fotopoulos and B. Flynn, “Optimum antenna coil structure for inductive powering of passive RFID tags,” in *Proc. IEEE Int. Conf. RFID*, Mar. 2007, pp. 71–77.
  - [39] G. P. Hancke and S. C. Centre, “Eavesdropping attacks on high-frequency RFID tokens,” in *Proc. Workshop RFID Security*, Jul. 2008, pp. 100–113.
  - [40] What is NFC Technology. [Online]. Available: <http://www.rfid-nfc.eu>
  - [41] E. Haselsteiner and K. Breitfuss, “Security in near field communication,” in *Proc. Workshop RFID Security*, Jul. 2006, pp. 3–13.
  - [42] H. Baldus, S. Corroy, A. Fazzi, K. Klabunde, and T. Schenk, “Human-centric connectivity enabled by body-coupled communications,” *IEEE Commun. Mag.*, vol. 47, no. 6, pp. 172–178, Jun. 2009.
  - [43] K. B. Rasmussen, C. Castelluccia, T. S. Heydt-Benjamin, and S. Capkun, “Proximity-based access control for implantable medical devices,” in *Proc. ACM Conf. Comput. Commun. Security*, Nov. 2009, pp. 410–419.
  - [44] D. Donoho, “Compressed sensing,” *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.
  - [45] M. Zhang, M. M. Kermani, A. Raghunathan, and N. K. Jha, “Energy-efficient and secure sensor data transmission using compression,” in *Proc. Int. Conf. VLSI Design*, Jan. 2013, pp. 31–36.
  - [46] Federal Information Processing Standards (FIPS), *Announcing the Advanced Encryption Standard (AES)*, Nov. 2001. [Online]. Available: <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
  - [47] National Institute of Standards and Technology (NIST), *SHA-3 Selection Announcement*. [Online]. Available: [http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3\\_selection\\_announcement.pdf](http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3_selection_announcement.pdf)
  - [48] T. Moore, “IEEE 802.11-01/610r02: 802.1x and 802.11 key interactions,” Microsoft Res., Tech. Rep., 2001.
  - [49] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik, “Radio-telepathy: Extracting a secret key from an unauthenticated wireless channel,” in *Proc. ACM Int. Conf. Mobile Comput. Netw.*, 2008, pp. 128–139.
  - [50] S. Jana et al., “On the effectiveness of secret key extraction from wireless signal strength in real environments,” in *Proc. Int. Conf. Mobile Comput. Netw.*, 2009, pp. 321–332.
  - [51] S. T. Ali, V. Sivaraman, and D. Ostry, “Zero reconciliation secret key generation for body-worn health monitoring devices,” in *Proc. ACM Conf. Security Privacy Wireless Mobile Netw.*, 2012, pp. 39–50.
  - [52] F. Xu, Z. Qin, C. C. Tan, B. Wang, and Q. Li, “IMDGuard: Securing implantable medical devices with the external wearable guardian,” in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2011, pp. 1862–1870.
  - [53] C. Cornelius et al., “Who wears me? Bioimpedance as a passive biometric,” in *Proc. USENIX Workshop Health Security Privacy*, Aug. 2012.
  - [54] S. Schechter, “Security that is meant to be skin deep: Using ultraviolet micropigmentation to store emergency-access keys for implantable medical devices,” Microsoft Res., Tech. Rep. MSR-TR-2010-33, Apr. 2010.
  - [55] T. Denning, K. Fu, and T. Kohno, “Absence makes the heart grow fonder: New directions for implantable medical device security,” in *Proc. Conf. Hot Topics Security*, Jul. 2008, pp. 1–7.
  - [56] S. Gollakota, H. Hassanieh, B. Ransford, D. Katabi, and K. Fu, “They can hear your heartbeats: Non-invasive security for implantable medical devices,” in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2011.
  - [57] M. Zhang, A. Raghunathan, and N. K. Jha, “MedMon: Securing medical devices through wireless monitoring and anomaly detection,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 7, no. 6, pp. 871–881, Dec. 2013.
  - [58] K. S. Kwak, S. Ullah, and N. Ullah, *An Overview of IEEE 802.15.6 Standard*, 2011. [Online]. Available: [arxiv.org/pdf/1102.4106](http://arxiv.org/pdf/1102.4106)
  - [59] P. Li and R. Bashirullah, “A wireless power interface for rechargeable battery operated medical implants,” *IEEE Trans. Circuits Syst.*, vol. 54, no. 10, pp. 912–916, Oct. 2007.
  - [60] P. D. Mitcheson, E. M. Yeatman, G. K. Rao, A. S. Holmes, and T. C. Green, “Energy harvesting from human and machine motion for wireless electronic devices,” *Proc. IEEE*, vol. 96, no. 9, pp. 1457–1486, Sep. 2008.
  - [61] *Infected Apps Found in Google’s Android Market*, 2011. [Online]. Available: <http://www.securitynewsdaily.com>
  - [62] iPhone malware paradigm, 2012. [Online]. Available: <http://www.crosstalkonline.org>
  - [63] S. W. Smith and S. Weingart, “Building a high-performance, programmable secure coprocessor,” *Comput. Netw.*, vol. 31, pp. 831–860, Apr. 1999.
  - [64] ARM, *Trustzone Technology Overview*. [Online]. Available: <http://www.arm.com/products/CPU/arch-trustzone.htm>
  - [65] Trusted Computing Group. [Online]. Available: <https://www.trustedcomputinggroup.org>
  - [66] N. Aaraj, A. Raghunathan, and N. K. Jha, “Analysis and design of a hardware/software trusted platform module for embedded systems,” *ACM Trans. Embedded Comput. Syst.*, vol. 8, pp. 8:1–8:31, Jan. 2009.
  - [67] J. M. Sorber, M. Shin, R. Peterson, and D. Kotz, “Plug-n-Trust: Practical trusted sensing for mHealth,” in *Proc. Int. Conf. Mobile Syst. Appl. Services*, 2012, pp. 309–322.
  - [68] J. Sorber et al., “An amulet for trustworthy wearable mHealth,” in *Proc. Workshop Mobile Comput. Syst. Appl.*, 2012, pp. 7:1–7:6.
  - [69] C. Cornelius et al., “Who wears me? Bioimpedance as a passive biometric,” in *Proc. USENIX Workshop Health Security Privacy*, Aug. 2012.
  - [70] C. Cornelius and D. Kotz, “Recognizing whether sensors are on the same body,” in *Proc. Int. Conf. Perv. Comput.*, 2011, pp. 332–349.
  - [71] N. Aaraj, A. Raghunathan, and N. K. Jha, “Virtualization-based framework for malware defense,” in *Proc. Conf. Detection Intrusions Malware Vulnerability Assessment*, Jul. 2008, pp. 64–87.
  - [72] D. Arora, S. Ravi, A. Raghunathan, and N. K. Jha, “Secure embedded processing through hardware-assisted run-time monitoring,” in *Proc. Design Autom. Test Eur. Conf.*, Mar. 2005, vol. 1, pp. 178–183.
  - [73] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in *Proc. Int. Cryptology Conf.*, Aug. 1999, pp. 388–397.
  - [74] P. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, other systems,” in *Proc. Int. Cryptology Conf.*, Aug. 1996, vol. 1109, pp. 104–113.
  - [75] D. Agrawal, B. Archambeault, and J. R. Rao, “The EM side-channel(s),” in *Proc. Workshop Cryptogr. Hardware Embedded Syst.*, Aug. 2002, pp. 29–45.
  - [76] D. Kune et al., “Ghost talk: Mitigating EMI signal injection attacks against analog sensors,” in *Proc. IEEE Symp. Security Privacy*, May 2013.
  - [77] M. A. Hasan, “Power analysis attacks and algorithmic approaches to their countermeasures for Koblitz curve cryptosystems,” *IEEE Trans. Comput.*, vol. 50, no. 10, pp. 1071–1083, Oct. 2001.
  - [78] K. Tiri, M. Akmal, and I. Verbauwhede, “A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards,” in *Proc. Eur. Solid-State Circuits Conf.*, 2002, pp. 403–406.
  - [79] K. Tiri and I. Verbauwhede, “Charge recycling sense amplifier based logic: Securing low power security ICs against DPA,” in *Proc. Eur. Solid-State Circuits Conf.*, 2004, pp. 179–182.
  - [80] R. Muresan and S. Gregori, “Protection circuit against differential power analysis attacks for smart cards,” *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1540–1549, Nov. 2008.
  - [81] G. B. Ratanpal, R. D. Williams, and T. N. Blalock, “An on-chip signal suppression countermeasure to power analysis attacks,” *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 3, pp. 179–189, Jul.-Sep. 2004.
  - [82] P.-C. Liu, H.-C. Chang, and C.-Y. Lee, “A low overhead DPA countermeasure circuit based on ring oscillators,” *IEEE Trans. Circuits Syst.*, vol. 57, no. 7, pp. 546–550, Jul. 2010.
  - [83] S. Yang, W. Wolf, N. Vijaykrishnan, D. N. Serpanos, and Y. Xie, “Power attack resistant cryptosystem design: A dynamic voltage and frequency switching approach,” in *Proc. IEEE Int. Symp. Circuits Syst.*, 2005, pp. 64–69.
  - [84] M. Bucci, R. Luzzi, M. Guglielmo, and A. Trifiletti, “A countermeasure against differential power analysis based on random delay insertion,” in *Proc. IEEE Int. Symp. Circuits Syst.*, 2005, pp. 3547–3550.
  - [85] M. Zhang and N. K. Jha, “FinFET-based power management for improved DPA resistance with low overhead,” *ACM J. Emerging Technol. Comput. Syst.*, vol. 7, no. 3, pp. 10:1–10:16, Aug. 2011.

## ABOUT THE AUTHORS

**Meng Zhang** received the B.S. degree in electronics from Beijing University, Beijing, China, in 2008 and the M.A. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 2010, where he is currently working toward the Ph.D. degree in electrical engineering.

His research interests include low-power system design, computer security, and body area networks.



**Anand Raghunathan** (Fellow, IEEE) received the B.Tech. degree in electrical and electronics engineering from the Indian Institute of Technology, Madras, India and the M.A. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, USA.

He is a Professor in the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. Previously, he was a Senior Research Staff Member at NEC Laboratories America, Princeton, NJ, USA, where he led research projects related to system-on-chip architectures, design methodologies, and heterogeneous parallel computing. He has coauthored a book *High-Level Power Analysis and Optimization* (New York, NY, USA: Springer-Verlag, 1998) and eight book chapters, holds 21 U.S. patents, and has presented several invited talks and conference tutorials. He has published over 200 journal and conference papers, and received eight best paper awards and four best paper nominations.

Prof. Raghunathan has been a member of the technical program and organizing committees of several leading conferences and workshops. He has chaired the ACM/IEEE International Symposium on Low Power Electronics and Design, the IEEE VLSI Test Symposium, the IEEE International Conference on VLSI Design, and the ACM/IEEE International Conference on Compilers, Architecture, and Synthesis for Embedded Systems. He has served as an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, *ACM Transactions on Design Automation of Electronic Systems*, the IEEE TRANSACTIONS ON MOBILE COMPUTING, *ACM Transactions on Embedded Computing Systems*, IEEE DESIGN & TEST OF COMPUTERS, and the *Journal of Low Power Electronics*. He was a recipient of the IEEE Meritorious Service Award (2001) and Outstanding Service Award (2004). He is a Golden Core Member of the IEEE Computer Society. He received a Patent of the Year Award (an award recognizing the invention that has achieved the highest impact), and two Technology Commercialization Awards from NEC. He was chosen by MIT's *Technology Review* among the TR35 (top 35 innovators under 35 years, across various disciplines of science and technology) in 2006, for his work on "making mobile secure."



**Niraj K. Jha** (Fellow, IEEE) received the B.Tech. degree in electronics and electrical communication engineering from Indian Institute of Technology, Kharagpur, India, in 1981, the M.S. degree in electrical engineering from the State University of New York (SUNY) at Stony Brook, Stony Brook, NY, USA, in 1982, and the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 1985.

He is a Professor of Electrical Engineering at Princeton University, Princeton, NJ, USA. He has coauthored or coedited five books titled *Testing and Reliable Design of CMOS Circuits* (Norwell, MA, USA: Kluwer, 1990), *High-Level Power Analysis and Optimization* (Norwell, MA, USA: Kluwer, 1998), *Testing of Digital Systems* (Cambridge, U.K.: Cambridge Univ. Press, 2003), *Switching and Finite Automata Theory* (Cambridge, U.K.: Cambridge Univ. Press, 2009, 3rd ed.), and *Nanoelectronic Circuit Design* (New York, NY, USA: Springer-Verlag, 2010). He has also authored 12 book chapters. He has authored or coauthored more than 390 technical papers. He has coauthored 14 papers, which have won various awards. He has received 14 U.S. patents. His research interests include FinFETs, low-power hardware/software design, computer-aided design of integrated circuits and systems, digital system testing, quantum computing, and secure computing. He has given several keynote speeches in the area of nanoelectronic design and test.

Prof. Jha is a Fellow of the Association for Computing Machinery (ACM). He has served as the Editor-in-Chief of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS and an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: REGULAR PAPERS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: EXPRESS BRIEFS, the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, and the *Journal of Electronic Testing: Theory and Applications*. He is currently serving as an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS, the *Journal of Low Power Electronics*, and the *Journal of Nanotechnology*. He has also served as the Program Chairman of the 1992 Workshop on Fault-Tolerant Parallel and Distributed Systems, the 2004 International Conference on Embedded and Ubiquitous Computing, and the 2010 International Conference on VLSI Design. He has served as the Director of the Center for Embedded System-on-a-chip Design funded by the New Jersey Commission on Science and Technology. He is the recipient of the AT&T Foundation Award and the NEC Preceptorship Award for research excellence, the NCR Award for teaching excellence, and the Princeton University Graduate Mentoring Award. His publications won the Best Paper Award at ICCD'93, FTCS'97, ICVLSID'98, DAC'99, PDCS'02, ICVLSID'03, CODES'06, ICCD'09, and CLOUD'10. A paper of his was selected for "The Best of ICCAD: A collection of the best IEEE International Conference on Computer-Aided Design papers of the past 20 years," two papers by IEEE MICRO MAGAZINE as one of the top picks from the 2005 and 2007 Computer Architecture conferences, and two others as being among the most influential papers of the last ten years at IEEE Design Automation and Test in Europe Conference. He has coauthored another six papers that have been nominated for best paper awards. He has served on the program committees of more than 140 conferences and workshops.



# Mobile Trusted Computing

*This paper surveys the trusted computing features in mobile computing platforms.*

By N. ASOKAN, JAN-ERIK EKBERG, KARI KOSTIAINEN, ANAND RAJAN, CARLOS ROZAS, AHMAD-REZA SADEGHI, STEFFEN SCHULZ, AND CHRISTIAN WACHSMANN

**ABSTRACT** | Trusted computing technologies for mobile devices have been researched, developed, and deployed over the past decade. Although their use has been limited so far, ongoing standardization may change this by opening up these technologies for easy access by developers and users. In this survey, we describe the current state of trusted computing solutions for mobile devices from research, standardization, and deployment perspectives.

**KEYWORDS** | Attestation; low-cost trusted execution; mobile security; physically unclonable functions (PUFs); trusted computing

## I. INTRODUCTION

The term trusted computing is used to collectively describe technologies enabling the establishment of trust in local and remote computing systems by using trustworthy components, trust anchors, to ensure the integrity of other parts of the system. In this paper, we survey recent research and industry efforts in designing and deploying trusted computing solutions, especially in the context of mobile systems.

Security in general and trusted computing technologies in particular have had a very different trajectory in the history of mobile devices compared to that of personal computers [54]. Various stakeholders had strict security requirements, some of which date back two decades ago,

right at the beginning of the explosion of personal mobile communications. For example, standards specifications required ensuring that the device identifier resists manipulation and change [1]; regulatory guidance called for secure storage for radio-frequency parameters calibrated during manufacture; business requirements necessitated ways of ensuring that subsidy locks<sup>1</sup> cannot be circumvented.

These requirements incentivized mobile device manufacturers, chip vendors, and platform providers to deploy hardware and platform security mechanisms for mobile platforms from early on. Hardware-based trusted execution environments (TEEs) were seen as essential building blocks in meeting these requirements. A TEE is a secure, integrity-protected processing environment, consisting of processing, memory, and storage capabilities. It is isolated from the “normal” processing environment, sometimes called the rich execution environment (REE) [39], where the device operating system and applications run. The term “rich” refers to the extensive functionality and, hence, the increased attack surface, in mass market operating systems today. TEEs enable improved security and usability for REE applications by ensuring that sensitive operations are restricted to the TEE and sensitive data, such as cryptographic keys, never leave the TEE.

The academic research community has been engaged in research in hardware-based trusted computing, although not using that specific term, for a long time dating back to the 1970s [5], [106]. Recent research efforts have focused on investigating alternative architectures for trusted computing, developing novel trust anchors using physically unclonable functions (PUFs), and enabling TEEs for resource-constrained devices [21], [97], [98].

Some of the results of these research activities have led to implementation proposals and large-scale deployment via standardization bodies and industry efforts. The Trusted Computing Group (TCG) [101] has been leading the standardization efforts in trusted computing. Global Platform [39] is specifying TEE functionality in mobile devices. Various application-specific standardization bodies, such as the Car Connectivity Consortium [67],

Manuscript received September 2, 2013; revised June 3, 2014; accepted June 13, 2014. Date of publication July 15, 2014; date of current version July 18, 2014.

**N. Asokan** is with the Intel Collaborative Research Institute for Secure Computing, University of Helsinki & Aalto University, Aalto FI-00076, Finland (e-mail: asokan@acm.org).

**J.-E. Ekberg** is with Trustonic, Aalto FI-00076, Finland (e-mail: jan-erik.ekberg@trustonic.com).

**K. Kostiainen** is with ETH Zurich, Zurich 8092, Switzerland (e-mail: kari.kostiainen@inf.ethz.ch).

**A. Rajan** and **C. Rozas** are with Intel Labs, Hillsboro, OR 97124-5961 USA (e-mail: anand.rajan@intel.com; carlos.v.rozas@intel.com).

**A.-R. Sadeghi** is with the Technische Universität Darmstadt, Darmstadt 64289, Germany, and also with the Center for Advanced Security Research Darmstadt (CASED), Darmstadt D-64293, Germany (e-mail: ahmad.sadeghi@trust.cased.de).

**S. Schulz** and **C. Wachsmann** are with the Intel Collaborative Research Institute for Secure Computing, Technische Universität Darmstadt, Darmstadt D-64293, Germany (e-mail: steffen.schulz@trust.cased.de; christian.wachsmann@trust.cased.de).

Digital Object Identifier: 10.1109/JPROC.2014.2332007

0018-9219 © 2014 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See [http://www.ieee.org/publications\\_standards/publications/rights/index.html](http://www.ieee.org/publications_standards/publications/rights/index.html) for more information.

<sup>1</sup>A subsidy lock prevents mobile phones subsidized by a mobile operator from being used by subscribers of a different mobile operator.

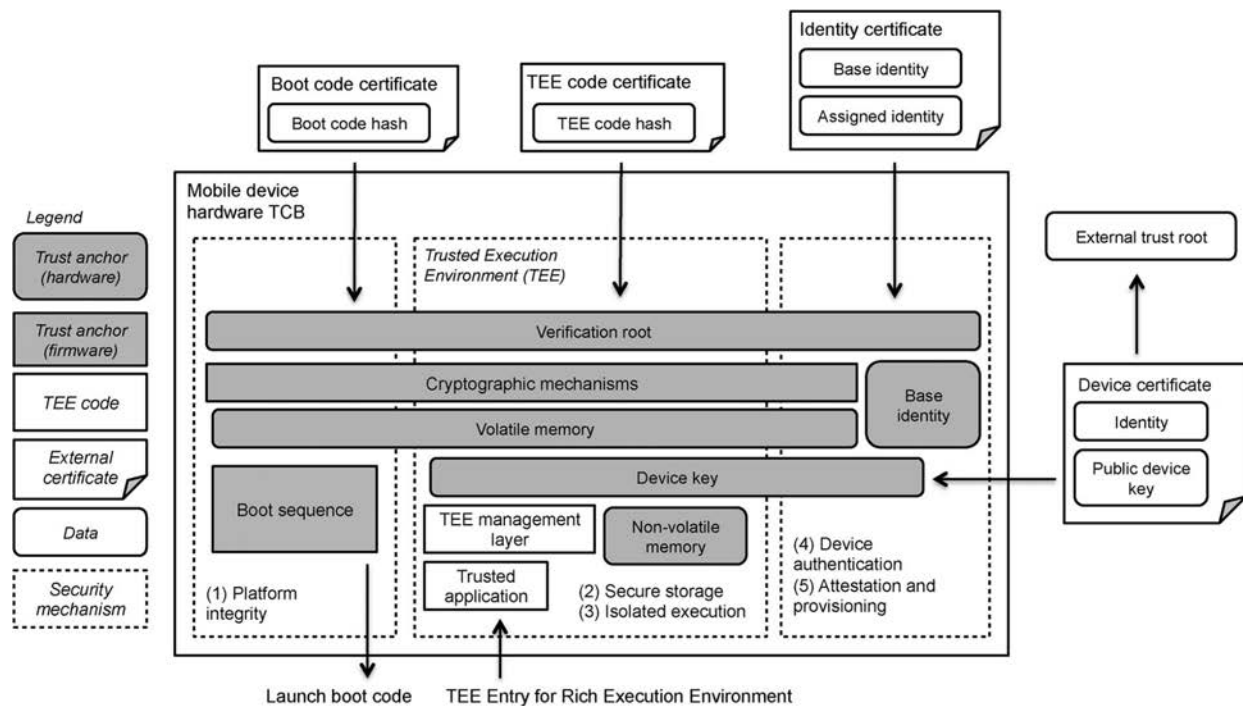


Fig. 1. Common hardware security concepts in mobile devices (adapted from [27]).

are formulating specifications for using trusted computing technologies to address security and privacy problems in specific application areas.

The first mobile phones with hardware-based TEEs appeared almost a decade ago [96]. A common way to realize a TEE in mobile devices is implementing a secure processor mode. An example of such an implementation is ARM TrustZone [9], which is present in smartphones and tablets today. Despite TEE deployment in many of these application areas, there has been no widely available means for application developers to benefit from existing TEE functionality apart from research or proprietary efforts [53].

With emerging standardization, this situation is about to change [27]. In the near future, we expect to see implementations of standardized interfaces for accessing and using TEEs emerging across different platforms. By making trusted computing technologies widely accessible, such a change will spur further research into novel trusted computing technologies and their applications in mobile and embedded devices.

Our goal in writing this survey is to explain the state of trusted computing solutions for mobile devices from research, standardization, and deployment perspectives. While we focus on mobile computing platforms, we also discuss technologies and approaches, such as lightweight trust establishment mechanisms and PUFs, which are relevant to mobile systems. We begin by outlining basic trusted computing concepts and introducing terminology

(Section II). We then discuss recent research (Section III), outline current and forthcoming new standard specifications (Section IV), and discuss various solutions proposed and deployed by the industry (Section V). Finally, we conclude by taking a step back and providing a perspective for the outlook of trusted computing technologies in mobile devices, outlining open issues (Section VI).

## II. BASIC CONCEPTS

The trusted computing base (TCB) of a mobile device consists of hardware and firmware components that need to be trusted unconditionally. In this survey, we denote such hardware and firmware components as trust anchors of the computing system.

Fig. 1, adapted from [27], illustrates trust anchors present in a typical mobile device. Individual trust anchors are shown in gray. The numbered dotted boxes (1–5) represent common security mechanisms and illustrate the trust anchors needed to implement each mechanism. In Sections II-A1–II-A5, we describe the security mechanisms. We use **bold font** whenever we introduce a concept shown in the figure for the first time.

### A. Basic Security Mechanisms

1) *Platform Integrity*: The integrity of platform code (e.g., the device OS) can be verified either during system boot or at device runtime. This allows device manufacturers and

platform providers to prevent or detect usage of platform versions that have been modified without authorization. Two variations of boot time integrity verification are possible.

In secure boot, the device startup process is stopped if any modification of the launched platform components is detected. A common approach to implement secure boot is to use code signing combined with making the beginning of the boot sequence immutable by storing it within the TCB (e.g., in ROM of the mobile device processor chip) during manufacturing [8]. The processor must unconditionally start executing from this memory location. Boot code certificates that contain hashes of booted code, signed with respect to a verification root, such as the device manufacturer public key stored on the device, can be used to verify the integrity of the booted components. The mobile device must be enhanced with cryptographic mechanisms to validate the signature of the system component launched first (e.g., the boot loader) that can in turn verify the next component launched (e.g., the OS kernel) and so on. If any of these validation steps fail, the boot process is aborted. Integrity of the cryptographic mechanisms can be ensured by storing the needed algorithms in ROM. The immutable boot sequence and a verification root together with an integrity-protected cryptographic mechanism provide the needed trust anchors for secure booting.

In authenticated boot, the started platform components are measured but not verified with respect to any reference values. Instead these measurements are logged in integrity-protected volatile memory. The boot loader measures the first component launched which in turn measures the next one and so on. The recorded measurements represent the state of the platform components after boot, and can be used for local access control enforcement or remote attestation (cf., Section II-A5). Two trust anchors are used to implement authenticated boot: integrity-protected volatile memory and a cryptographic mechanism.

Boot time integrity alone is not sufficient if an attacker can modify the system after it has been booted. In runtime platform integrity verification, a trusted software (or firmware) component monitors the integrity of the platform code continuously [76] and repairs modified components automatically if possible [50]. The integrity of the monitor itself can be verified using the above described boot integrity verification techniques.

2) *Secure Storage*: A mechanism to store data on the device to disallow unauthorized access by REE components is called secure storage. Sensitive data kept in secure storage should not leak to an attacker even if the REE is compromised. A common way to implement secure storage is to augment the device hardware configuration with a confidential and integrity-protected device-specific key that can be accessed only by authorized code. Such a device key may be initialized during manufacturing and stored in a

protected memory area on the processor chip. To protect against key extraction by physical attacks, manufacturing techniques like protective coatings may be used. In addition to the device key, implementation of secure storage requires trusted implementations of necessary cryptographic mechanisms, such as an authenticated encryption algorithm. Data rollback protection requires the inclusion of writable nonvolatile memory (e.g., a monotonic counter) that persists its state across device boots.

To summarize, two trust anchors are needed for secure storage: a device key and cryptographic mechanisms. Note that securely storing cryptographic keys is useful only if cryptographic algorithms using these keys are protected as well.

3) *Isolated Execution*: The term “isolated execution” refers to the ability to run security-critical code outside the control of the untrusted REE. Isolated execution combined with secure storage constitutes a TEE, which allows implementation of various security applications that resist REE compromise. We explain possible TEE architectures in Section II-B. Here, we introduce the trust anchors needed to implement a TEE, which are a subset of the mobile device hardware TCB. Conceptually, the TEE can be seen as a component of the TCB.

A TEE can expose the functionality of predefined cryptographic mechanisms to the REE with the guarantee that the cryptographic keys never leave the TEE. While predefined common cryptographic operations are sufficient for many services, certain applications require isolated execution of application-specific algorithms. Proprietary one-time password algorithms for online banking constitute one such example. To support isolated execution of arbitrary code, the device hardware configuration must provide an interface (TEE entry) through which the executable code (trusted applications) can be loaded for execution using the protected volatile memory.

A TEE code certificate can authorize code execution within the TEE and authorize trusted applications to access the device key and other device resources such as confidential data (e.g., digital rights management keys) and hardware interfaces (e.g., the cellular modem or near-field communication interface). Furthermore, the access that any trusted application has to the device key and other device resources may be controlled based on the platform state that was measured and saved during an authenticated boot process.

A software or firmware component called TEE management layer provides a runtime environment for trusted applications and enforces access control to protected resources like the device key (more details in Section II-B). The integrity of the management layer must be verified either as part of the boot time platform integrity verification (and runtime monitoring) or on demand when trusted applications are loaded for execution [64]. Realization of isolated execution can make use of the following trust

anchors: isolated memory (volatile or nonvolatile), cryptographic mechanisms, and verification root.

4) *Device Authentication*: An external service provider can use device authentication to verify the identity of the mobile device (and its TEE). The identity may include device manufacturer information that can imply compliance to external service provider requirements.

The mobile device hardware configuration typically has a unique immutable base identity, which may be a serial number from a managed namespace or a statistically unique identifier initialized randomly at manufacture. A combination of a verification root and the base identity allows flexible device identification. An identity certificate that is signed with respect to the aforementioned verification root binds an assigned identity to the base identity. International mobile equipment identifier (IMEI) and link-layer identities such as Bluetooth and WiFi addresses are examples of device identities.

A device certificate signed by the device manufacturer can bind any assigned identity to the public part of the device key. Signatures over device identities using the device key provide device authentication toward external verifiers.

5) *Attestation and Provisioning*: An externally verifiable statement about the software configuration running on a device is called remote attestation. Remote attestation allows an external service provider to verify that a device is running a compliant platform version. A common way to implement remote attestation is to provide statements signed with the certified device key over authenticated measurements (e.g., cryptographic hash digests) of the firmware and software components loaded at boot time.

The process of securely sending secrets and code to the TEE of the target device is called provisioning. Many security services require a security association between an external service provider and the TEE of the correct user device. For example, a bank might want to provision a key to the TEE of a customer device for online banking authentication. In some cases, service providers also need to provision TEE code that operates on the provisioned secrets, such as proprietary one-time password algorithms. Device authentication provides the basis for TEE provisioning. Data can be provisioned encrypted under a certified device key. Device certificates do not include user identities and thus provisioning user authentication must be implemented by other means.

Note that all cryptographic keys needed for secure storage, isolated execution, device authentication, attestation, and provisioning can be derived from the same device key.

## B. TEE Architecture

The isolation needed for a TEE can be realized in various ways, ranging from separate security elements to

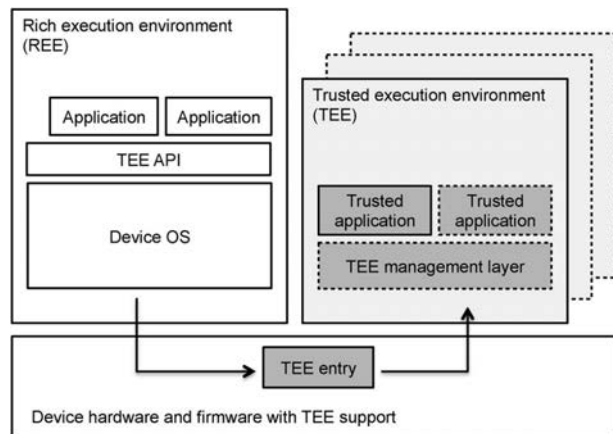


**Fig. 2. Generic TEE architecture model (adapted from [39]).** *Trusted applications are executed within a TEE instance that is isolated from the REE device OS. One or more TEE instances may be supported. Dashed boxes illustrate entities that are not present in all TEE architectures; gray boxes are not controlled by the REE device OS.*

secure processor modes and virtualization. Depending on the used isolation technique, different TEE architectures are possible. Fig. 2, adapted from [39], depicts a generic, high-level TEE architecture model that applies to different TEE architecture realizations.

We call a processing environment that is isolated from the REE device OS as TEE instance. A TEE architecture realization may support one or more TEE instances. In TEE architectures that are based on dedicated security chips [102] and processor modes [9], [96], typically a single TEE instance is available. Virtualization [63] and emerging processor architectures [65], [72] are TEE examples in which each REE application may create its own TEE instance. TEE instances are created (or activated) and accessed using the TEE entry interface. Applications running in the REE device OS access TEE services through a TEE application programming interface (API) that allows REE applications to execute trusted applications and to read and write data to and from them.

If only a single TEE instance is available, the same TEE instance typically allows execution of multiple trusted applications. The TEE management layer can be implemented in software as a full-fledged embedded OS, a set of libraries, a small interpreter that runs within the TEE or in device hardware and firmware. It provides the interface through which trusted applications communicate with REE applications and invoke cryptographic operations within the TEE. In terms of size and complexity, the management layer is likely to be significantly smaller than the REE device OS, and thus, its attack surface is smaller. In TEE architectures, where each REE application creates its own TEE instance, a management layer may not be used.

### III. RESEARCH SOLUTIONS

In the following, we discuss various concepts and research efforts that continue to extend and improve trusted computing research. Observe that we mainly focus on the recent research directions in mobile and embedded systems, while trusted computing following the approach used by the TCG is covered in Section IV. A detailed survey on research in traditional trusted computing is available in [75].

#### A. Alternative Trusted Computing Designs

One of the earliest works that describe the use of secure coprocessors to assure isolated execution (cf., Section II-A3) is the report on the 4758 Secure Coprocessor [24]. It describes the design of a physically isolated, tamper-resilient execution environment which implements a TEE that communicates with the central processing unit (CPU) to execute certain tasks securely on a separate processor and memory [108]. Following this work, it was investigated how remote parties can gain assurance that a particular application has been executed in the TEE of some particular device (cf., Section II-A5). A trust chain was devised by which the TEE itself can vouch for the execution of a particular code, which in turn may load and execute other code [95]. The device key of the TEE is embedded by its manufacturer, who vouches for the correct operation of that TEE.

Drawbacks of secure coprocessors are the high additional costs and the generally low computation performance. Copilot [76] alleviates this problem by using the coprocessor only to monitor and assure the integrity of the actual computation performed by the main CPU. Overshadow [20] uses hardware-assisted virtualization to enforce different views on memory for user applications and OS kernels, thus ensuring the integrity and confidentiality of applications despite OS compromise.

Some works have also investigated the extension of the CPU itself to enable the measurement of executing code and to establish a TEE. For instance, the AEGIS system architecture [99] extends the CPU interface with facilities for loading, measuring, and authenticating software modules, and uses these facilities to provide authenticated execution of tasks in real (nonvirtual) memory. Similarly, it was proposed that a CPU vendor could provide trusted software modules (TSMs) [23]. The code segments of TSMs are extended with authentication codes which are automatically verified when they are loaded into the cache banks of a CPU during execution.

Leveraging such a trusted loader or regular secure/authenticated boot (cf., Section II-A1), a minimal security kernel can be launched which then in turn ensured a measured and isolated execution of software tasks. In particular, the PERSEUS system architecture [77] proposes to leverage a secure microkernel for strong isolation between a multitude of software security services. The next-generation secure computing base (NGSCB) [29] proposes

an ultimately trusted security kernel to support secure applications in a secure world mode, while Terra [34] argues that a chain of trust must be established from platform initialization to the hypervisor and the individual executed applications. Trusted hypervisors such as sHype [85] and TrustVisor [63] follow this design and use a minimal security kernel that provides strong isolation between higher layer applications.

#### B. Remote Attestation

Remote attestation (cf., Section II-A5) begins with the initial measuring of the bootloader and OS [48], [91]. Integrity measurement architecture (IMA) [44], [86] extends the Linux kernel with facilities to measure loaded code and data according to predefined policies. During attestation, the software measurements maintained by the kernel can then be signed by the device key (cf., Section II-A2) and the kernel in turn can be verified based on the measurements performed by the bootloader and platform firmware. As an alternative, secure OS kernels such as PERSEUS or TrustVisor only measure certain isolated security services, which are then used by regular applications to perform secure transactions on their behalf [63], [77]. The security services are designed to provide maximum flexibility while maintaining low internal complexity and external dependencies, thus simplifying the process of measuring, validating, and establishing trust in a particular software [3], [88].

When extending a secure channel protocol with remote attestation, care must be taken that the reported measurements actually originate from the particular platform that is to be attested [40]. Multiple works have proposed protocol extensions for secure channels such as SSL and IPsec [10], [88] and extend the resulting networks into security domains of assured distributed access control enforcement (e.g., [17] and [62]).

A general problem in remote attestation is the disclosure of the often privacy-sensitive software state to the verifying entity (verifier). To address the problems of privacy but also scalability when dealing with large amounts of software integrity measurements, property-based attestation [19], [83] proposes to attest concrete properties of software. For this purpose, the loaded software is equipped with property certificates which ensure that the software has certain properties. During attestation, the platform then proves the existence of the required properties of the loaded software to the verifier. However, the identification and extraction of the desired software security properties from software remains an open problem [70].

#### C. Low-Cost Trusted Execution Environments

With the rise of resource-constrained embedded systems as part of complex monitoring and control infrastructures, a recent line of research investigates the possibility to perform attestation (cf., Section II-A5) and isolated execution (cf., Section II-A3) even on such

low-end devices. These works typically assume that common approaches like secure coprocessors or complex CPU modes are too expensive in terms of production cost or energy consumption. Instead, they aim to provide a limited trusted computing functionality for the purpose of automated verification and trust establishment in larger IT infrastructures.

1) *Software-Based Attestation*: If the mobile device does not support a hardware-protected key needed for remote attestation (as described in Section II-A5), attestation can be implemented in software. A typical software-based attestation scheme exploits the computational constraints of a device to make statements about its internal software state [92], [93]. The prover must compute a response to a given attestation challenge within a certain time. When receiving the correct response in the expected time, the verifier has assurance that only a specific attestation algorithm could have been executed within that time frame. The attestation algorithm is typically implemented as a specific checksum function that iteratively merges information gathered from the device. A formal analysis of software-based attestation [12] has shown the challenges of formalizing the underlying assumptions.

Several variations and extensions to software-based attestation have been proposed, ranging from implementations for different platforms to more fundamental changes to the software-based attestation concept, such as repeated challenge-response procedures [45], [58] or using memory constraints [33], [104], and self-modifying or obfuscated algorithms to prevent manipulation of the attestation algorithm [37], [92], [94]. Multiple works consider the combination of software-based attestation with hardware trust anchors such as TPMs [55], [87] and SIM cards [45] to authenticate the prover device.

2) *Minimal Attestation Hardware*: The secure minimal architecture for root of trust (SMART) [21] is designed to enable remote attestation and isolated execution at the lowest possible hardware cost (see also [31]). SMART realizes this using a custom access control enforcement on the memory bus, allowing access to a particular secret key in memory only if the current CPU instruction pointer (IP) points to a known trusted code region in ROM (secure storage). This way, the secret key is only accessible when the CPU is executing that trusted code and can thus be used to authenticate the execution of that ROM code to other parties. In particular, by letting the trusted ROM code measure and execute arbitrary code, the design can be extended to a freely programmable trusted execution mechanism or simply be used to attest the local platform.

While SMART is more efficient and easier to validate than software-based attestation, it suffers from certain practical drawbacks. In particular, SMART offers no exception or interrupt handling, requiring a platform reset and memory clearing in case of unexpected errors. To

prevent interruption of the trusted code, the hardware access control in SMART assures that the corresponding code region can only be entered at the first address and exited at its last address. However, memory protection based on the CPU instruction pointer may still be exploited with code reuse attacks, where the semantics of code is changed based on stack or other data manipulation [30].

3) *CPU-Based Task Protection*: Another approach to isolated execution and possibly low-cost trusted execution are self-protected modules (SPMs) [98]. They extend the CPU instructions to provide trusted execution based on execution-dependent memory protection, allowing tasks to request protected memory regions and query the protection status of other tasks in the system directly from the CPU. This way, protected tasks can inspect and attest each other in local memory. For communication and multitasking, protected tasks can declare code entry points which may be called by other tasks with the desired arguments, while other portions of code are protected by the platform. However, when communicating with other tasks on the local platform, one needs to assure that the other task's entry points and protection status have not been changed since the last local attestation.

Sancus [72] extends an openMSP430 CPU to implement SPMs in hardware. However, the problem of handling interrupts and unexpected software faults remains unsolved, and additional modifications are required to sanitize the platform memory upon device reset. To assure to local tasks that a particular other task has not been modified (e.g., by malware), the CPU provides a number of cryptographic tokens and secure hashes of individual loaded tasks. As a result, Sancus imposes relatively high hardware costs for the targeted low-end systems, imposing a 100% area increase for providing eight secure modules [72]. By managing tasks through CPU instructions, Sancus imposes certain restrictions on the memory layout of a task, e.g., limiting capabilities for shared memory or peripherals input/output (I/O). Another implementation of SPMs is provided in the Fides hypervisor [97]. Fides can provide secure interruption and communication between processes, which, however, seems to be achievable also with typical task isolation by trusted hypervisors or security kernels.

4) *Execution-Aware Memory Protection*: TrustLite [51] extends the concepts of SMART [21] and SPM [98] to provide a programmable, execution-aware memory protection subsystem for low-cost embedded devices. TrustLite's execution-aware memory protection unit (EA-MPU) allows running a number of protected tasks (trustlets) in parallel without requiring additional CPU instructions. Moreover, the EA-MPU can be programmed to provide individual trustlets with shared memory and exclusive peripherals access, enabling the construction of secure device drivers and other platform services.

TrustLite also proposes a modified CPU exception engine to prevent information leakage to OS interrupt handlers. This allows the OS to perform preemptive multitasking of trustlets similar to regular tasks, thus facilitating integration of trustlets with existing software stacks.

To address the assumption of SMART and Sancus that all system memory is cleared on platform reset, TrustLite deploys a secure loader that initializes the EA-MPU at boot time, thus allowing an efficient reallocation and protection of sensitive memory regions prior to REE invocation. Additionally, instead of having the hardware managing identification tokens for secure interprocess communication (IPC) as in Sancus, TrustLite assumes that low-cost embedded systems do not require the reallocation or upgrade of TEE tasks at runtime but that TEEs can remain in memory until platform reset.

#### D. Physically Unclonable Functions

In many scenarios, emerging low-cost devices are exposed to physical attacks. Thieves or even rightful users may attempt to extract cryptographic keys, to clone the device or to manipulate its software. However, protecting secure storage (cf., Section II-A2) against hardware attacks requires the integration of expensive physical security mechanisms that are not economical for low-cost devices. In this context, PUFs represent a promising new security primitive, which enables unique device identification and authentication [78], [84], binding software to hardware [25], [56], secure storage of cryptographic secrets [59], and remote attestation protocols [90].

1) *PUF Concept, Properties, and Assumptions*: A PUF is a physical object, e.g., an integrated circuit [60] that, when queried with a challenge, generates a response which depends on both the challenge and the unique device-specific physical properties of the PUF. PUFs are typically assumed to be robust, unclonable, unpredictable, and tamper evident [11]. Informally, robustness means that, when queried with the same challenge multiple times, the PUF returns a similar response with high probability. Unclonability demands that it is infeasible to produce two PUFs that cannot be distinguished based on their challenge-response behavior. Unpredictability requires that it is infeasible to predict the PUF response to an unknown challenge, even if the PUF can be adaptively queried for a certain number of times. A PUF is tamper evident if any attempt to physically access the PUF irreversibly changes its challenge-response behavior.

Since PUFs are affected by operating conditions, such as ambient temperature variations, they return slightly different responses when queried with the same challenge multiple times. Furthermore, PUF responses are not uniformly random. Hence, PUFs are typically combined with fuzzy extractors [22], which map similar PUF responses to the same value (error correction) and extract full-entropy bit strings from the PUF response (privacy amplification).

2) *PUF Types*: There is a variety of PUF implementations (see [60] and [82] for an overview). The most appealing ones for the integration into electronic circuits are electronic PUFs, which come in different flavors. Delay-based PUFs are based on race conditions or frequency variations in integrated circuits and include arbiter PUFs [57] and ring oscillator PUFs [36]. Memory-based PUFs exploit the instability of volatile memory elements, such as SRAM cells [41], flip-flops [103], and latches [56].

3) *PUF-Based Device Authentication*: Device authentication (cf., Section II-A4) typically relies on a secret key securely stored in the device. While classical approaches to secure storage (cf., Section II-A2) may be too expensive or even technically infeasible for resource-constrained embedded devices, PUFs promise to provide a lightweight alternative to secure device authentication. The most common approach [78] of using PUFs for device authentication is that the device manufacturer stores a set of challenge-response pairs (CRPs) in a database which can later be used by a verifier to identify the device. However, a general problem of this approach is that CRPs cannot be reused since this would enable replay attacks. A more practical approach is based on standard authentication protocols and stores the authentication secret in a PUF-based key storage (cf., Section III-D4).

4) *Secure Key Generation and Storage*: Classical approaches to secure storage (cf., Section II-A2) are often not suitable for low-cost embedded systems. In this context, PUFs can be used to securely bind secrets (such as cryptographic keys) to a device. Instead of storing the key in secure nonvolatile memory, the key is extracted from the physical properties of the underlying hardware each time it is used [25], [56]. This protects the key against unauthorized readout by invasive attacks, such as probing attacks against nonvolatile memory. Moreover, when using a tamper-evident PUF, any attempt to physically extract the key changes the PUF and securely deletes the key.

5) *PUF-Based Remote Attestation*: Software-based attestation (cf., Sections II-A5 and III-C1) implies that, due to the lack of secure storage, cryptographic schemes that rely on secrets cannot be used. However, software-based attestation assumes that the prover device is authenticated to the verifier, which is hard to achieve without using cryptographic authentication. To overcome this problem, the attestation algorithm executed by the prover must be linked to the hardware it was computed on, which can be achieved by using PUFs [89], [90]. To assure that the attestation algorithm is not outsourced to another device, the constraints of the communication interfaces of the prover are exploited similar to the way the computational constraints of the prover are exploited by standard software-based attestation. Due to the uniqueness of the PUF responses and their tight integration into the

attestation algorithm, a correct and timely attestation response provides assurance of the identity of a remote device as well as on the integrity of its software state.

6) *Security of PUF-Based Solutions*: In contrast to most cryptographic primitives, whose security can be related to well-established intractability assumptions, the security of PUFs relies on physical properties. Many PUF security models exist (see [11] for an overview) that, however, do not capture the properties of real PUF implementations, which can lead to attacks on PUF-based systems [81]. A PUF security framework that aims to capture the properties of real PUF implementations and that allows for empirically assessing and quantifying these properties for PUF implementations has been presented in [11]. A large-scale security analysis of application-specific integrated circuit (ASIC) implementations of the most popular electronic PUF types [13], [47] shows that PUF implementations are sufficiently robust but not all of them achieve the desired security properties (e.g., unpredictability).

Most known implementations of PUFs can be emulated in software [57], [80], either by reading them out completely or by model building attacks. The complexity of these attacks can be increased by obfuscating the actual PUF response [35], [61]. However, this requires protecting the implementation of the algorithms obfuscating the PUF response against invasive and side-channel attacks. Research on the side-channel analysis of PUF-based systems has recently started [46], [66], [74].

## IV. STANDARDIZATION

Industry standards consortia have recently intensified efforts to standardize TEE functionality and its interfaces. Standardization aims at agreeing on common APIs for provisioning and trustworthy execution across devices and software ecosystems and the ability to subject the TEE to compliance and security certification. In this section, we provide a brief overview of relevant standards dealing with TEE functionality in a bottom-up manner.

Standards that define services which make use of TEE functionality are important as well. Examples include Car Consortium [67] and MobeyForum [68]. Due to lack of space, we do not discuss these standards further.

### A. National Institute of Standards and Technology

The U.S. National Institute of Standards and Technology (NIST) draft 800.164 [18] provides definitions and terminology for many aspects of mobile hardware security ranging from ownership roles to policy enforcement and usage scenarios like “bring your own device” (BYOD). But its most significant contribution is its unified categorization for roots of trust (RoTs), which is NIST’s term for hardware trust anchors.

NIST guidelines clearly and concisely collect and describe RoTs for reporting, verification, storage, and mea-

surement. In particular, NIST identifies a new RoT for integrity, which was not previously discussed in other work. This RoT represents the isolated environment where measurements and trusted state assertions can be securely stored when the device is active.

The RoTs are one way to agree on a hardware security foundation for TEEs. Each RoT can be evaluated and graded according to the level of security it can provide to the system. More importantly, as we mentioned in Section II, RoTs are the abstract tools on which the main capabilities of a TEE system are built: isolation, secure storage, and integrity.

### B. Global Platform (GP)

The Global Platform Device Specifications, in particular the architecture document [39], have established the reference model and terminology used for TEE software architectures we introduced in Section II. The trusted applications (TAs) that run in the TEE are written using the TEE internal API [38], which is the reference library interface for TAs. The internal API includes interfaces for cryptographic functions, secure storage, and I/O. Especially the parameter passing paradigm between the REE device OS and TAs is a significant divergence from what typically is available for traditional secure elements such as smart cards. A GP TA gets its input and provides its output using memory references residing in the address space of the caller in the REE device OS. This allows TAs to access and process client-side memory directly, e.g., for in-place decryption or the processing of vast client-side data structures, say, for runtime integrity checking.

### C. Mobile Hardware Security APIs

Some mobile platforms provide APIs for hardware-assisted cryptographic operations. Java MicroEdition, widely used in feature phones, defines JSR 177 [73] as a generic smartcard-like interface which can be used to expose a standard cryptographic API (the implementation may be provided by a mobile phone secure element such as a SIM card). Recent versions of the Android platform expose an API for hardware-assisted cryptography [28] in the form of a standard PKCS 11 interface [79], while in iOS similar functionality is provided through a proprietary API [7].

These hardware-security APIs have been modeled after usage paradigms of hardware security modules (HSMs), cryptographic tokens, and smart cards. A traditional hardware security API allows creation of hardware-protected keys and common cryptographic operations, such as encryption and signatures, using these keys. To take advantage of the programmability of mobile TEEs (isolated execution), a different kind of API abstraction is needed. The API should address provisioning of trusted applications and secrets into the device, authorization of trusted applications to access provisioned secrets and device keys, and control which REE application can execute trusted applications. None of the current standardized or

*de facto* proprietary hardware-security APIs provide such functionality.

#### D. Trusted Computing Group

The trusted platform module (TPM) [102], defined by the TCG, is a functional interface for platform security. TPMs are widely deployed today. A TPM contains functions for key generation and use, mainly with asymmetric encryption and signature primitives, but also for symmetric cryptography. All TPM processing happens in isolation from the REE OS and the caller, governed by the RoTs described in Section IV-A. Furthermore, TPMs provide primitives for data sealing and unsealing, monotonic counters, randomness, and some limited amount of non-volatile storage.

1) *Platform Configuration Registers (PCRs)*: One notable distinction of TCG specifications is that they provide platform binding as an inherent service for the authorization and attestation of TPM objects and functions. This feature sets it apart from other standards such as GP. All TPM implementations provide some number of PCRs which are integrity-protected writable volatile memory that reside within the TPM. PCRs are used to cryptographically aggregate successive measurements of software/hardware components or configurations originating from the REE. Aggregating a new measurement to a PCR is known as PCR extension and is realized using a cryptographic hash function. If the REE OS consistently provides measurements of all its code to the TPM, before executing the measured code, then the set of PCRs serve as a representation of the state of the currently running REE software. PCRs have two uses: binding TPM objects to REE state and reporting REE state to external verifiers as part of remote attestation (cf., Section II-A5). TPM objects such as keys and stored data can be associated with an assertion that restricts the use of the object only when the trusted platform is in a certain predefined state, expressed as set of predefined reference values for PCRs. Generating a signed report of PCR values with a certified TPM-specific key is the essential step in remote attestation of the REE state.

2) *TPM Mobile*: TPM mobile specifications [previously known as mobile trusted module (MTM)] allow a TPM to be realized in software within a TEE as a TA. This makes it possible to have more than one active TPM instance in a device. This multistakeholder model allows different stakeholders (such as device manufacturers, mobile operators, enterprises, and users) to run their “own” TPMs and populate their TPM with measurements relevant to their needs, independently of other TPMs active on the same system.

3) *TPM2 Authorization*: A new revision (v2.0) [102] of the TPM standards (TPM2) are currently on the road to publication. The new specifications make improvements in

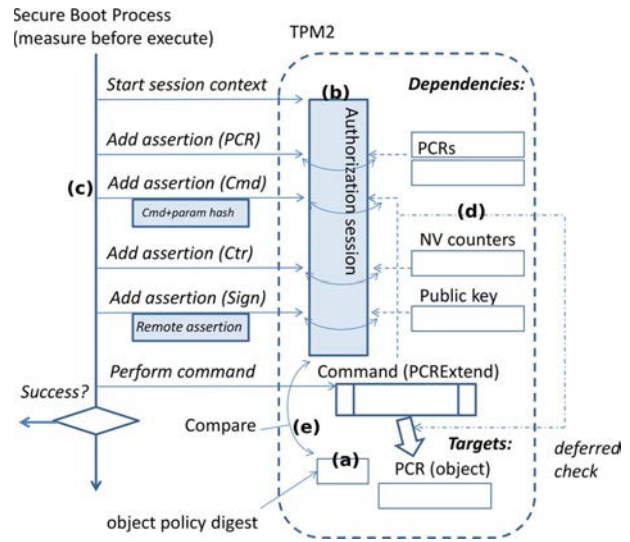


Fig. 3. Secure boot with TPM authorization.

various aspects, such as algorithm agility and the use of nonvolatile memory. The most novel aspect of TPM2 is its enhanced authorization model. In this section, we illustrate the power of this model by showing how it can be used for secure boot, which is a common need for mobile and embedded devices.

The guiding principle for the authorization model is separation of policy and mechanism [107]. This allows designing the secure boot of a platform from the OS upward in a hardware-independent manner, where any binary relation can be applied as a policy assertion. We will now explain an example secure boot policy depicted in Fig. 3.

In version 1.x of the TPM specifications, the means of expressing access control policies is mainly limited to passwords, and many object operations (e.g., extending PCRs) cannot be associated with a policy. For example, a device could conceivably implement secure boot by aggregating a sequence of measurements of launched software components in a PCR and aborting the boot process if the PCR does not reach an expected reference value. The reference value would, however, have to be encoded in data objects (certificates) outside the TPM. Naturally, this requires a rigid specification of the mechanism in order to make it apply across manufacturers and systems.

In the example in Fig. 3, we use TPM2 authorization to make secure boot conditioned on the successful completion of the TPM2 command PCRExtend to extend a specific PCR. We must, therefore, associate the policy for secure boot with the ability to perform this operation.

Most TPM2 objects can be associated with a single, unique object policy digest [cf., Fig. 3(a)], which is permanently stored with the object. In order to get authorization for object access, the caller must run an authorization session [cf., Fig. 3(b)] that contains a rollback-protected,

accumulated session policy digest. The session is updated using TPM policy commands [cf., Fig. 3(c)] that each implement a policy assertion, which consists of some system evidence [cf., Fig. 3(d)] that the TPM2 knows about. This evidence is aggregated into the session digest in a well-known and well-defined manner. Session policy digests are aggregated in the same way as PCRs are extended, i.e., by using a cryptographic hash function.

In our example session, the secure boot process makes three assertions at the beginning: the first one aggregates the current state of the system, represented by some source PCRs; the second assertion binds the exact target command (PCRExtend), along with its parameters such as the identity of the target PCR and the value it is going to be extended with; the third assertion binds the value of a counter (for example, specifying a bound for the current version number of the firmware being booted). After all these assertions have been aggregated, the session policy digest will end up with a specific value  $Q$ . The final assertion in our example (remote assertion) is an external signature on a structure asserting the expected value  $Q'$  of a session digest when all prior assertions have been completed. If this signature can be validated with a public key  $P$  loaded into the TPM and the session digest matches reference value  $Q = Q'$ , then the session digest is replaced with a unique representation of the public key<sup>2</sup> that validated the signature, say  $W = H(P)$ . If we had set the reference object policy for the target PCR [cf., Fig. 3(a)] to be the same value  $W$ , then issuing the PCRExtend command in the context of the authorization session will succeed as long as both the digests match [cf., Fig. 3(e)] and deferred checks on conditions implied by the provided assertions hold.<sup>3</sup> If the policy values do not match or the deferred assertions fail, then the PCRExtend invocation returns with a failure code. Based on this, the caller should take appropriate action, like resetting the device.

In other words, this process effectively allows having a remotely controlled authorization policy where an external signature controls which policies are allowed. The remote controller can issue signed remote assertions for multiple different session digests, each representing an alternate acceptable configuration for secure boot. TPM2 also supports ways of securely mapping accumulated session policy digests to perform logical access control decisions.

One example is the PolicyOR command. On the condition that the current policy digest value matches one of a number of values in a set provided as a PolicyOR param-

eter, then the current digest value is replaced with a representation of the entire set of values.

All the assertions listed above exist in the published TPM2 command set: a secure boot process along these lines is viable with a TPM2. As the caller cannot affect the contribution of any policy assertion, the only way the authorization can succeed is that the state described by the assertions actually is the one represented in the external signature, thus achieving separation of mechanism and policy.

Note that TPM2 is a passive entity. To implement secure boot using TPM2, there has to be an external immutable active entity, such as a boot loader, which enforces secure boot. It is customary for such secure boot implementations to allow the user to override secure boot [15].

## V. INDUSTRY SOLUTIONS

Over the past years, several trusted computing research concepts have been realized in industry products, and in many cases such products have fostered new opportunities to build and research trusted systems. In the following sections, we review some of the main technologies as well as standardization efforts.

### A. Virtualization and Dynamic Root of Trust

Many mobile and ultramobile laptop platforms feature hardware-assisted virtualization technology, such as Intel virtualization technology (Intel VT). A central design goal of Intel VT was to simplify the implementation of robust hypervisors. Intel VT adds two new operation modes: VMX root mode for hypervisors and VMX nonroot mode for virtual machines. VMX root mode is very similar to the normal Intel architecture without Intel VT while VMX nonroot mode provides an Intel architecture environment controlled by a hypervisor. A virtual-machine control structure (VMCS) was introduced to facilitate transitions between VMX root mode and VMX nonroot mode and can be programmed by the hypervisor to establish boundaries on a VM, including access to memory, devices, and control registers. While operating in VMX nonroot mode, the execution of certain instructions and events causes a transition to VMX root mode called a VMexit. The hypervisor can retrieve details as to the cause of the VMexit by reading the VMCS and process the event accordingly [71]. Intel VT introduced a generalized IO-MMU architecture which enables system software to define constraints on direct memory access (DMA) devices, restricting their access to specific subsets of physical memory allowing for a smaller TCB [2].

Another major capability of modern systems is the dynamic root of trust for measurement (DRTM). Available as Intel trusted execution technology (Intel TXT) or AMD secure virtual machine, this technique enables a CPU to perform a runtime reinitialization and establish a new software TCB (TEE payload), irrespective of the

<sup>2</sup>The public key used to verify the signature assertion is loaded into the TPM from the outside by the secure boot process. However, the TPM will extend the used public key into the session digest which ensures that the subsequent command will succeed only if the correct public key was used to verify the signature.

<sup>3</sup>Some assertions, like, e.g., the target command assertion, cause a deferred check to be logged in the session and validated during actual command execution in addition to the update of the session digest at the time the assertion is made.

trustworthiness of previously loaded software. For this purpose, the TCG TPM was extended with a set of DRTM PCRs which can be reset at runtime by the CPU by sending a TPM command from the appropriate operation mode (TPM locality). The Intel GETSECS [SENDER] instruction initiates the DRTM. The CPU resets the DRTM PCRs and loads an authenticated code module (ACM) into an isolated execution environment. The ACM performs a series of platform configuration checks, configures DMA protection for the TEE payload, and extends the TEE payload hashes into the TPM PCRs.

DRTM technology has been used to securely execute critical software payloads such as SSH logins, X.509 e-mail signatures, or to protect banking secrets [16], [32], [64]. Intel TXT has also been used in combination with Intel VT to initiate a trusted hypervisor, which in turn provides multiple TEEs to the individual running VMs [63]. The generalized IO-MMU allows hypervisors to be “disengaged,” i.e., to only perform an initial configuration of VM boundaries, thus providing only a minimal external interface and complexity [49]. Alternatively, a “disaggregated” hypervisor may reduce its TCB by delegating drivers for peripherals control to other VMs [69], or to construct a trusted path, providing secure user I/O for TEEs [110].

## B. Userspace Trusted Execution

Intel software guard extensions (Intel SGX) are a set of new instructions and memory access changes to the Intel architecture to support TEEs. The extensions provide the ability to instantiate one or more TEEs (enclaves) that reside within an application inside an REE. Accesses to the enclave memory area against software (not resident in the enclave) are prevented by hardware. This restriction is enforced even from privileged software, such as operating systems, virtual machine monitors, and the basic input/output system (BIOS).

The enclave lifecycle begins when a protected portion of an application is loaded into an enclave by system software. The loading process measures the code and data of the enclave and establishes a protected linear address range for the enclave. Once the enclave has been loaded, it can be accessed by the application as a service or directly as part of the application. On first invocation, the enclave can prove its identity to a remote party and be securely provisioned with keys and credentials. To protect its data persistently, the enclave can request a platform-specific key unique to the enclave to encrypt data and then use untrusted services of the REE.

To implement Intel SGX memory protections, new hardware and structures are required. The enclave page cache (EPC) is a new region of protected memory where enclave pages and structures are stored. Inside the EPC, code and data from many different enclaves can reside. The processor maintains security and access control information for every page in the EPC in a hardware structure called the enclave page cache map (EPCM). This structure

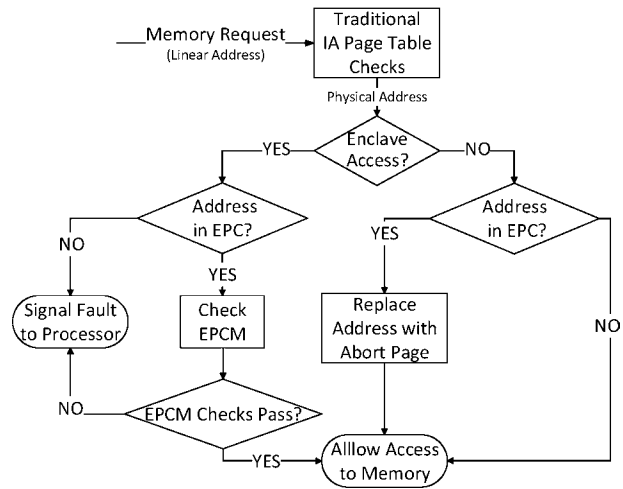


**Fig. 4. High-level architecture of Intel SGX PMH. Processor memory requests are sent to the PMH for translation and access control checks. As part of Intel SGX, the PMH has been extended to check whether a memory access was initiated by an enclave. For nonenclave accesses, the PMH redirects any access to the EPC to nonexistent memory (abort page). For an enclave access (an access by enclave to its protected linear address range), the PMH checks that the translated address is an EPC page. Furthermore, the PMH consults the EPCM to verify that the EPC page belongs to enclave requesting access, the correct linear address was used to access the page, and access permissions are consistent with the request.**

is consulted by the processor’s page miss handler (PMH) hardware module, as shown in Fig. 4. The PMH mediates access to memory by consulting page tables maintained by system software, range registers, and the EPCM. A memory encryption engine (MEE) protects the EPC when using main memory for storage [65].

Enclave binaries are loaded into the EPC using new instructions. ECREATE starts the loading process and initializes the Intel SGX enclave control structure (SECS) which contains global information about the enclave. EADD loads a page of content into a free EPC page and records the commitment into the SECS. Once the EPC page has been loaded, the contents of the page are measured using EEXTEND. After all the contents of the enclave have been loaded into the EPC, EINIT completes the creation process by finalizing the enclave measurement and establishes the enclave identity. Until an EINIT is executed, enclave entry is not permitted.

Once an enclave has been loaded, it can be invoked by application software. To enter and exit an enclave programmatically (e.g., as part of a call/return sequence), new instructions, EENTER and EEXIT, are provided. While operating in enclave mode, an interrupt, fault, or exception may occur. In this case, the processor invokes a special internal routine called asynchronous exit (AEX) which saves and clears the enclave register state and translation lookaside buffer (TLB) entries for the enclave. The

ERESUME instruction restores the processor state to allow the enclave to resume execution.

To enable attestation and sealing, the hardware provides two additional instructions EREPORT and EGETKEY. The EREPORT instruction provides an evidence structure that is cryptographically protected using symmetric keys. EGETKEY provides enclave software with access to keys used in the attestation and sealing process. A special quoting enclave is devoted to remote attestation. The quoting enclave verifies REPORTs from other enclaves on the platform and creates a signature using a device specific (private) asymmetric key [4].

Intel SGX minimizes the TCB of trusted applications to the critical portion of the application and hardware. This simplifies the creation and validation of remote attestation reports, as remote verifiers no longer have to understand multiple TEE management layers and their dependencies. While requiring CPU extensions, Intel SGX does not require any dependencies on the TPM, a hypervisor, or a separate trusted operating system. Further, it is protected against hardware and software attacks on RAM. Finally, Intel SGX enables application developers to directly deploy trusted applications inside REE applications [42].

### C. Mobile Architectures Using Secure Processor Modes

ARM TrustZone [9] and TI M-Shield [96] are system-wide, mobile security architectures that leverage a secure execution mode provided by the device main CPU. TrustZone is deployed to the majority of current smartphones, whereas M-Shield is an example of a previous, similar architecture. In this section, we focus on TrustZone.

The main CPU of the mobile device supports two execution modes called normal world and secure world. The processor boots to the secure world which sets up the necessary runtime environment before switching to the normal world. Execution can switch back to the secure world when a special command is executed in the normal world. This command starts a monitor mode that performs the processor mode switch. The designer of a mobile device hardware configuration defines the hardware components that are accessible in these two modes.

Fig. 5 illustrates an example hardware configuration in a TrustZone-enabled mobile device [9]. The device main CPU, small amounts of RAM and ROM, and the cellular modem are integrated into a system on chip (SoC). These on-chip components are connected with an internal bus. The SoC also includes memory controllers for off-chip memory elements and peripherals.

The access control between these hardware elements is implemented using a status flag that the SoC internal bus carries. The status flag indicates the mode of the master device in bus communication. Bus communication slaves must enforce access control based on the flag. Hardware elements can be made aware of the status flag or dedicated access control hardware can be placed be-

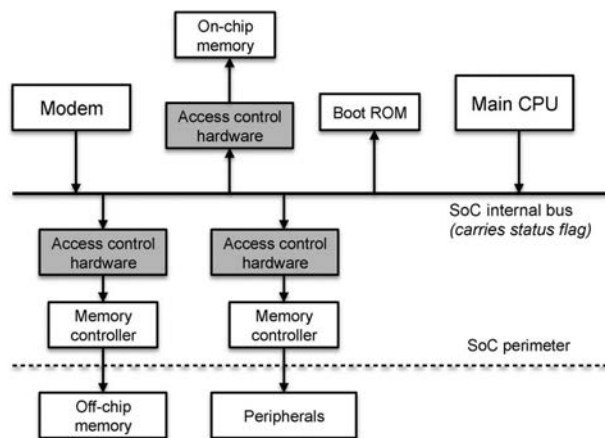


**Fig. 5. Overview of the ARM TrustZone system architecture [9].** In a typical mobile device, many hardware elements are integrated into a single SoC. Access control between TrustZone normal world and secure world can be implemented with a system bus flag and dedicated access control hardware (gray boxes).

tween the bus and the target hardware element. Access to memory elements and peripherals is typically controlled by adding dedicated access control hardware elements between the bus and the hardware element or its memory controller.

Typically, on-chip memory is configured for secure world access only, while the off-chip memory elements and peripherals can be partitioned between the secure world and the normal world. Most peripherals are accessible by the normal world. Also interrupt handling can be configured; the processor can switch execution mode for dedicated interrupts, if needed.

Typical uses of TrustZone TEE include secure boot, secure storage, and isolated execution. The underlying trust anchors (device key, cryptographic mechanism, and verification root) can be configured during manufacturing into on-chip memory. The same trust anchors can be used for device authentication and attestation.

### D. Secure Elements

Besides processor modes, smartphones support TEEs in the form of separate security elements, such as smart cards. Some mobile devices have extension slots for dedicated smart card TEEs and the majority of mobile devices are equipped with a SIM card. Smart card TEEs provide secure storage and isolated execution. Boot integrity verification, device authentication, and remote attestation are typically not supported by smart card TEE realizations.

### E. Onboard Credentials

Although TEE architectures like TrustZone and M-Shield have been deployed to smartphones for almost a decade, and secure elements in the form of SIM cards are present in many mobile devices, the usage of

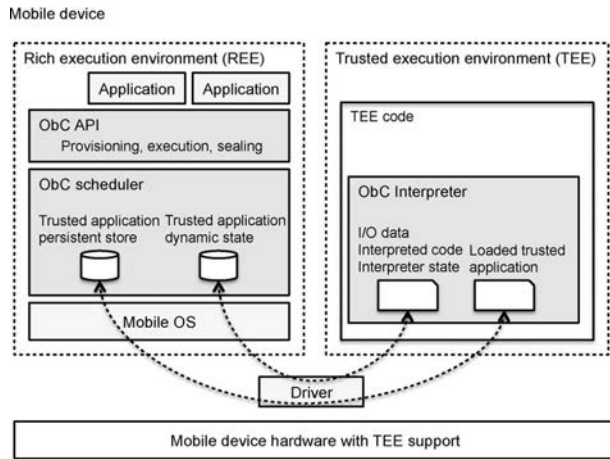


**Fig. 6. Overview of ObC architecture [27]. ObC interpreter is implemented as TEE code. ObC scheduler controls execution of dynamically loaded ObC trusted applications and maintains persistent state. Mobile applications access ObC trusted applications through ObC API that provides provisioning, execution, and sealing functionality.**

hardware-security mechanisms by third-party developers has been limited [27]. Traditionally, mobile device manufacturers have leveraged mobile TEE capabilities only for their internal use cases, such as subsidy lock protection or secure boot and secure element application development has not been available for third parties.

Onboard credentials (ObC) [26], [52], [53] is a TEE architecture developed at Nokia Research Center and currently available in TrustZone-enabled Nokia Windows Phone 8 and Symbian phones. The ObC system serves as an example of TEE architecture that allows developers to utilize programmability of mobile TEEs.

Fig. 6 illustrates the ObC architecture [27]. Trusted applications are executed on top of a small virtual machine called the ObC interpreter (TEE management layer). The interpreter provides isolation between trusted applications originating from unknown developers. Trusted application development can be done in BASIC or using bytecode assembler. The ObC interpreter is implemented as a set of interacting TEE code components. Depending on the underlying TEE hardware the components may permanently reside inside the TEE or be loaded on demand.

To execute a trusted application, the ObC scheduler loads along with its inputs and stored data sealed by previous invocations. The ObC interpreter executes the trusted application bytecode. Some execution events will cause the interpreter to collect its runtime state, encrypt it, and return to the REE for scheduling. The ObC scheduler reinvokes the same or different trusted applications, attaching possible temporarily stored data or the interpreter state, and in this manner the bytecode execution continues to completion. Numerous context switches cause a significant execution overhead, but since the system runs on

the mobile device main CPU, the achieved performance is comparable to solutions without scheduling on slower security chips, such as smart cards.

The ObC platform supports an open provisioning model in which any developer can, with the permission of the device user, deploy trusted applications. A device-specific and manufacturer-certified public key provides the basis for remote provisioning; service providers need to handle user authentication additionally. The certified device key can transport a provisioner-specific secret that defines a new security domain. Isolation between security domains inside the TEE is guaranteed by interleaving execution of different security domains in time and implementing local storage with distinct encryption keys.

To develop a complete security service, a service provider needs to deploy a trusted application that handles the service-specific security logic within the TEE, and a REE application that triggers the trusted application execution (and provides a user interface).

Smartphone-based public transport ticketing is an example ObC application that has been deployed in practice [100]. In nongated transport systems, travel tickets are not verified at station gates, but instead travelers are requested to perform ticketing on their own accord but are subject to ticket inspections. Such a model allows a traveller to stop his phone from reporting evidence for trips during which he was not inspected. An ObC trusted application that implements an authenticated counter bound to identity verification signatures can address such situations. A traditional cryptographic API (e.g., PKCS11 [79] or TPM interface [102]) would not enable implementation of authenticated counters. With a programmable TEE, implementation of such a ticketing solution is simple, and deployment to devices already in the field is practical.

## F. Physically Unclonable Functions

While PUFs and PUF-based security solutions are still investigated by the research community, security products based on PUFs are already announced for the market [43], [105]. These systems mainly target IP-protection and anti-counterfeiting applications as well as secure key storage and device authentication systems.

## VI. OUTLOOK AND SUMMARY

The role of trusted computing features in mobile and embedded devices is at a crossroads. After years of limited use, the imminent arrival of new standards and increased interest on the part of the industry to make trusted computing widely accessible has the potential to increase and change the ways by which application developers and users benefit from these features. The fact that industry sees new opportunities in this domain is evident from the arrival of new products from established companies and the formation of new companies. With increased use of mobile devices, new research problems will become apparent.

On the other hand, several issues need to be addressed before such increased use becomes reality. We discuss some of them below.

*Privacy and trust issues:* With the improved scalability and security of modern TEEs, previously raised concerns [6] regarding privacy and vendor lock-in are becoming more important. For instance, the implications of combining TPM2 with secure boot are currently subject of intensive discussions in Europe [14], [109]. This calls for further research in industry and academia to analyze and improve the security and privacy implications of this emerging technology.

*Future of PUFs as trust anchor:* There are many yet unsolved challenges with regard to the scalable and secure integration of PUFs into IT systems. For instance, most existing PUF implementations can be emulated in software and require to obfuscate the actual PUF responses to prevent emulation attacks. In particular, side-channel attacks and invasive hardware attacks can be used to extract PUF responses. Hence, further research should investigate alternative PUF designs that are resistant to emulation attacks and/or the secure integration of the PUF and the logic processing the PUF responses.

*Attacker models:* Hardware security solutions need to consider a number of attack vectors such as side-channel attacks based on memory management and cache manipulation, power consumption analysis, pipeline analysis, or interface timing attacks. While protections against such threats are known, they are expensive and unsuitable for low-cost devices. The impact of possible attacks can be mitigated by suitable system designs. For example, TEE implementations that make use of chip-external memory effectively extend the TEE across multiple components within a device. Such TEEs are vulnerable to memory probing attacks using physical probes. If an application protected by such TEEs is designed to avoid the use of global keys (keys shared by all devices in the system) or “class keys” (keys shared by a large group of devices), then the impact of a successful attack on a single device can be minimized.

*Mitigation of software attacks in hardware:* One big challenge is the verification of the runtime integrity of an

IT system. Existing approaches to detect and to prevent control flow attacks (such as return-oriented programming attacks) are typically implemented in software and involve a significant runtime overhead. An important line of research, therefore, is to investigate how mechanisms to protect the control flow integrity of IT systems can be effectively realized in hardware to achieve higher performance.

*Provisioning:* As discussed in Section V-E, the ability for developers to freely provision trusted applications and credentials subject only to user approval (and without necessarily requiring approval from any third parties like device manufacturers or mobile operators) has the potential to rapidly expand uses of TEE. Architectures like ObC (Section V-E) and Intel SGX (Section V-B) facilitate such open provisioning. Global Platform is working on a new “consumer-centric provisioning model” with the same intent.

*Scaling down:* Just as the popularity and deployment of smartphones exploded during the last decade, the widespread use of low-cost devices in security and privacy-critical infrastructures may follow. Remote attestation and isolated execution are important features impacting the feasibility and scalability of these emerging systems. We discussed multiple proposed solutions in Section III-C, each with their own limitations and cost constraints. However, currently no comprehensive security architecture exists to facilitate the secure operation of low-cost embedded systems.

*Peripherals:* In addition to secure execution and storage within the TEE, it is necessary to secure their interactions with peripherals. How this can be done in a flexible, yet economic manner is an open question.

*Usability:* How trusted applications and credentials in a device can be securely and easily backed up or migrated from one device to another on user demand are critical to the usability of trusted computing technologies. Equally important is the consideration of “usability” for developers. What programming paradigms should be used for the development of trusted applications? ■

## REFERENCES

- [1] 3GPP, “3GPP TS 42.009 security aspects,” 2001. [Online]. Available: <http://www.3gpp.org/ftp/Specs/html-info/42009.htm>
- [2] D. Abramson et al., “Intel virtualization technology for directed I/O,” *Intel Technol. J.*, vol. 10, no. 3, pp. 179–192, 2006.
- [3] A. Alkassar, M. Scheibel, A.-R. Sadeghi, C. Stübke, and M. Winandy, “Security architecture for device encryption and VPN,” in *Information Security Solution Europe (ISSE)*. Berlin, Germany: Vieweg-Verlag, 2006.
- [4] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, “Innovative technology for CPU based attestation and sealing,” in *Hardware and Architectural Support for Security and Privacy (HASP)*. New York, NY, USA: ACM, 2013.
- [5] J. P. Anderson, “Computer security technology planning study,” U.S. Air Force, Tech. Rep. ESD-TR-73-51, 1972, vol. II. [Online]. Available: <http://csrc.nist.gov/publications/history/ande72.pdf>
- [6] R. Anderson, “Trusted computing: Frequently asked questions,” 2003. [Online]. Available: <http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html>
- [7] Apple Inc., “iOS security,” 2012. [Online]. Available: [http://images.apple.com/iphone/business/docs/iOS\\_Security\\_Oct12.pdf](http://images.apple.com/iphone/business/docs/iOS_Security_Oct12.pdf)
- [8] W. A. Arbaugh, D. J. Farber, and J. M. Smith, “A secure and reliable bootstrap architecture,” *Proc. IEEE Symp. Security Privacy*, 1997, pp. 65–71.
- [9] ARM, “ARM security technology—Building a secure system using TrustZone technology,” 2009. [Online]. Available: <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prtd29-genc-009492c/index.html>
- [10] F. Armknecht et al., “An efficient implementation of trusted channels based on OpenSSL,” in *Proc. ACM Workshop Scalable Trusted Comput.*, 2008, pp. 41–50.
- [11] F. Armknecht, R. Maes, A.-R. Sadeghi, F.-X. Standaert, and C. Wachsmann, “A formal foundation for the security features of physical functions,” in *Proc. IEEE Symp. Security Privacy*, 2011, pp. 397–412.
- [12] F. Armknecht, A.-R. Sadeghi, S. Schulz, and C. Wachsmann, “A security framework for the analysis and design of software attestation,” in *Proc. ACM Conf. Comput. Commun. Security*, 2013, DOI: 10.1145/2508859.2516650.
- [13] M. Bhargava, C. Cakir, and K. Mai, “Comparison of bi-stable and delay-based physical unclonable functions from

- measurements in 65 nm bulk CMOS,” in *Proc. IEEE Custom Integr. Circuits Conf.*, 2012, DOI: 10.1109/CICC.2012.6330625.
- [14] BMI, “Key requirements on ‘trusted computing’ and ‘secure boot,’” German Government white paper, 2012. [Online]. Available: [http://www.bmi.bund.de/SharedDocs/Downloads/DE/Themen/OED\\_Verwaltung/Informationsgesellschaft/trusted\\_computing\\_eng.html](http://www.bmi.bund.de/SharedDocs/Downloads/DE/Themen/OED_Verwaltung/Informationsgesellschaft/trusted_computing_eng.html)
- [15] J. Bottomley and J. Corbet, “Making UEFI secure boot work with open platforms,” Linux Foundation white paper, 2011. [Online]. Available: [https://www.linuxfoundation.org/sites/main/files/lf\\_uefi\\_secure\\_boot\\_open\\_platforms.pdf](https://www.linuxfoundation.org/sites/main/files/lf_uefi_secure_boot_open_platforms.pdf)
- [16] F. F. Brasser, S. Bugiel, A. Filyanov, A.-R. Sadeghi, and S. Schulz, “Softer smartcards—Usable cryptographic tokens with secure execution,” in *Financial Cryptography*. New York, NY, USA: Springer-Verlag, 2012.
- [17] S. Cabuk et al., “Towards automated security policy enforcement in multi-tenant virtual data centers,” *Comput. Security*, vol. 18, pp. 89–121, 2010.
- [18] L. Chen, J. Franklin, and A. Regenscheid, “Guidelines on hardware-rooted security in mobile devices,” National Institute of Standards Technology (NIST), Tech. Rep. SP 800-164. [Online]. Available: [http://csrc.nist.gov/publications/drafts/800-164/sp800\\_164\\_draft.pdf](http://csrc.nist.gov/publications/drafts/800-164/sp800_164_draft.pdf)
- [19] L. Chen, H. Löhr, M. Manulis, and A.-R. Sadeghi, “Property-based attestation without a trusted third party,” in *Proc. 11th Int. Conf. Inf. Security*, 2008, pp. 31–46.
- [20] X. Chen et al., “Overshadow: A virtualization-based approach to retrofitting protection in commodity operating systems,” *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 2, pp. 2–13, 2008.
- [21] K. E. Defrawy, A. Francillon, D. Perito, and G. Tsudik, “SMART: Secure and minimal architecture for (establishing a dynamic) root of trust,” in *Proc. Netw. Distrib. Syst. Security Symp.*, 2012, Internet Society.
- [22] Y. Dodis, J. Katz, L. Reyzin, and A. Smith, “Robust fuzzy extractors and authenticated key agreement from close secrets,” in *Advances in Cryptology (CRYPTO)*. New York, NY, USA: Springer-Verlag, 2006.
- [23] J. Dworkin and R. Lee, “Hardware-rooted trust for secure key management and transient trust,” in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 389–400.
- [24] J. Dyer et al., “Building the IBM 4758 secure coprocessor,” *IEEE Computer*, vol. 34, no. 10, pp. 57–66, Oct. 2001.
- [25] I. Eichhorn, P. Koeberl, and V. van der Leest, “Logically reconfigurable PUFs: Memory-based secure key storage,” in *Proc. 6th ACM Workshop Scalable Trusted Comput.*, 2011, pp. 59–64.
- [26] J.-E. Ekberg, “Securing software architectures for trusted processor environments,” Ph.D. dissertation, Dept. Comput. Sci. Eng., Aalto Univ., Aalto, Finland, 2013. [Online]. Available: <http://urn.fi/URN:ISBN:978-952-60-3632-8>
- [27] J.-E. Ekberg, K. Kostiainen, and N. Asokan, “The untapped potential of trusted execution environments on mobile devices,” 2014. *IEEE Security Privacy Mag.*, DOI: 10.1109/MSP.2014.38.
- [28] N. Elenkov, “Jelly bean hardware-backed credential storage,” 2012. [Online]. Available: <http://nelenkov.blogspot.ch/2012/07/jelly-bean-hardware-backed-credential.html>
- [29] P. England, B. Lampson, J. Manferdelli, M. Peinado, and B. Willman, “A trusted open platform,” *IEEE Computer*, vol. 36, no. 7, pp. 55–63, Jul. 2003.
- [30] A. Francillon, Q. Nguyen, K. B. Rasmussen, and G. Tsudik, “Systematic treatment of remote attestation,” 2012. [Online]. Available: <http://eprint.iacr.org/2012/713.pdf>
- [31] A. Francillon, Q. Nguyen, K. B. Rasmussen, and G. Tsudik, “A minimalist approach to remote attestation,” *Proc. Design Autom. Test Eur. Conf. Exhibit.*, 2014, DOI: 10.7873/DATE2014.257.
- [32] S. Gajek, H. Löhr, A.-R. Sadeghi, and M. Winandy, “TruWallet: Trustworthy and migratable wallet-based web authentication,” in *Proc. ACM Workshop Scalable Trusted Comput.*, 2009, pp. 19–28.
- [33] R. W. Gardner, S. Garera, and A. D. Rubin, “Detecting code alteration by creating a temporary memory bottleneck,” *IEEE Trans. Inf. Forensics Security*, vol. 4, no. 4, pp. 638–650, Dec. 2009.
- [34] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, “Terra: A virtual machine-based platform for trusted computing,” in *Proc. 19th ACM Symp. Oper. Syst. Principles*, 2003, pp. 193–206.
- [35] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, “Controlled physical random functions,” in *Proc. IEEE Annu. Comput. Security Appl. Conf.*, 2002, pp. 149–160.
- [36] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, “Silicon physical random functions,” in *Proc. 9th ACM Conf. Comput. Commun. Security*, 2002, pp. 148–160.
- [37] J. T. Giffin, M. Christodorescu, and L. Kruger, “Strengthening software self-checksumming via self-modifying code,” in *Proc. IEEE 21st Annu. Comput. Security Appl. Conf.*, 2005, DOI: 10.1109/CSAC.2005.53.
- [38] Global Platform, “TEE internal API specification,” 2011. [Online]. Available: <http://www.globalplatform.org/specificationsdevice.asp>
- [39] Global Platform, “TEE system architecture,” 2011. [Online]. Available: <http://www.globalplatform.org/specificationsdevice.asp>
- [40] K. Goldman, R. Perez, and R. Sailer, “Linking remote attestation to secure tunnel endpoints,” in *Proc. 1st ACM Workshop Scalable Trusted Comput.*, 2006, pp. 21–24.
- [41] J. Guajardo, S. Kumar, G.-J. Schrijen, and P. Tuyls, “FPGA intrinsic PUFs and their use for IP protection,” in *Cryptographic Hardware and Embedded Systems (CHES)*. New York, NY, USA: Springer-Verlag, 2007.
- [42] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade, and J. Del Cuvillo, “Using innovative instructions to create trustworthy software solutions,” *Proc. 2nd ACM Int. Workshop Hardware Architect. Support Security Privacy*, 2013, DOI: 10.1145/2487726.2488370.
- [43] Intrinsic ID, “Products,” 2013. [Online]. Available: <http://www.intrinsic-id.com/products.htm>
- [44] T. Jaeger, R. Sailer, and U. Shankar, “PRIMA: Policy-reduced integrity measurement architecture,” in *Proc. 11th ACM Symp. Access Control Models Technol.*, 2006, pp. 19–28.
- [45] M. Jakobsson and K.-A. Johansson, “Retroactive detection of malware with applications to mobile platforms,” in *Proc. USENIX Conf. Hot Topics Security*, 2010, article 1–13.
- [46] D. Karakoyunlu and B. Sunar, “Differential template attacks on PUF enabled cryptographic devices,” in *Proc. IEEE Int. Workshop Inf. Forensics Security*, 2010, DOI: 10.1109/WIFS.2010.5711445.
- [47] S. Katzenbeisser et al., “PUFs: Myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon,” in *Cryptographic Hardware and Embedded Systems (CHES)*. New York, NY, USA: Springer-Verlag, 2012.
- [48] B. Kauer, “OSLO: Improving the Security of Trusted Computing,” in *Proc. 16th USENIX Security Symp.*, 2007, article 16.
- [49] E. Keller, J. Szefer, J. Rexford, and R. B. Lee, “NoHype: Virtualized cloud infrastructure without the virtualization,” in *Proc. 37th Annu. Int. Symp. Comput. Architect.*, 2010, pp. 350–361.
- [50] M. S. Kirkpatrick, G. Ghinita, and E. Bertino, “Resilient authenticated execution of critical applications in untrusted environments,” *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 4, pp. 597–609, Jul./Aug. 2012.
- [51] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan, “Trustlite: A security architecture for tiny embedded devices,” in *Proc. Eur. Conf. Comput. Syst.*, 2014, DOI: 10.1145/2592798.2592824.
- [52] K. Kostiainen, “On-board credentials: An open credential platform for mobile devices,” Ph.D. dissertation, Dept. Comput. Sci. Eng., Aalto Univ., Aalto, Finland, 2012. [Online]. Available: <http://urn.fi/URN:ISBN:978-952-60-4598-6>
- [53] K. Kostiainen, J.-E. Ekberg, N. Asokan, and A. Rantala, “On-board credentials with open provisioning,” in *Proc. 4th ACM Symp. Inf. Comput. Commun. Security*, 2009, pp. 104–115.
- [54] K. Kostiainen, E. Reshetova, J.-E. Ekberg, and N. Asokan, “Old, new, borrowed, blue—A perspective on the evolution of mobile platform security architectures,” in *Proc. 1st ACM Conf. Data Appl. Security Privacy*, 2011, pp. 13–24.
- [55] X. Kovah et al., “New results for timing-based attestation,” in *Proc. IEEE Symp. Security Privacy*, 2012, pp. 239–253.
- [56] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, “Extended abstract: The butterfly PUF protecting IP on every FPGA,” in *Proc. IEEE Int. Workshop Hardware-Oriented Security*, 2008, pp. 67–70.
- [57] J. W. Lee et al., “A technique to build a secret key in integrated circuits for identification and authentication applications,” in *Proc. Symp. VLSI Circuits*, 2004, pp. 176–179.
- [58] Y. Li, J. M. McCune, and A. Perrig, “VIPER: Verifying the Integrity of PERipherals’ firmware,” in *Proc. 18th ACM Conf. Comput. Commun. Security*, 2011, pp. 3–16.
- [59] D. Lim et al., “Extracting secret keys from integrated circuits,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 10, pp. 1200–1205, Oct. 2005.
- [60] R. Maes and I. Verbauwhede, “Physically unclonable functions: A study on the state of the art and future research directions,” in *Towards Hardware-Intrinsic Security*. New York, NY, USA: Springer-Verlag, 2010.
- [61] M. Majzoobi, F. Koushanfar, and M. Potkonjak, “Lightweight secure PUFs,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2008, pp. 670–673.
- [62] J. M. McCune, S. Berger, R. Caceres, T. Jaeger, and R. Sailer, “Bridging mandatory access control across machines,” IBM Res. Div., Res. Rep. RC23778, 2005.

- [63] J. M. McCune et al., "TrustVisor: Efficient TCB reduction and attestation," in *Proc. IEEE Symp. Security Privacy*, 2010, pp. 143–158.
- [64] J. M. McCune, B. J. Parno, A. Perrig, M. K. Reiter, and H. Isozaki, "Flicker: An execution infrastructure for TCB minimization," in *Proc. 3rd ACM Eur. Conf. Comput. Syst.*, 2008, pp. 315–328.
- [65] F. McKeen et al., "Innovative instructions and software model for isolated execution," in *Proc. 2nd Int. Workshop Hardware Architect. Support Security Privacy*, 2013, DOI: 10.1145/2487726.2488368.
- [66] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Side-channel analysis of PUFs and fuzzy extractors," in *Trust and Trustworthy Computing (TRUST)*. New York, NY, USA: Springer-Verlag, 2011.
- [67] Car Connectivity Consortium. [Online]. Available: <http://www.mirrorlink.com/>
- [68] Mobey Forum, "Business models for NFC payments," 2011. [Online]. Available: <http://www.mobeyforum.org/business-models-for-nfc-payments-white-paper>
- [69] D. G. Murray, G. Milos, and S. Hand, "Improving Xen security through disaggregation," in *Proc. 4th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ.*, 2008, pp. 151–160.
- [70] A. Nagarajan, V. Varadarajan, M. Hitchens, and E. Gallery, "Property based attestation and trusted computing: Analysis and challenges," in *Proc. 3rd Int. Conf. Netw. Syst. Security*, 2009, pp. 278–285.
- [71] G. Neiger, A. Santoni, F. Leung, D. Rodgers, and R. Uhlig, "Intel virtualization technology: Hardware support for efficient processor virtualization," *Intel Technol. J.*, vol. 10, no. 3, pp. 167–177, 2006.
- [72] J. Noorman et al., "Sancus: Low-cost trustworthy extensible networked devices with a zero-software trusted computing base," in *Proc. 22nd USENIX Conf. Security*, 2013, pp. 479–494.
- [73] Java Community Process, "JSR 177: Security and Trust Services API for J2ME," 2007. [Online]. Available: <http://jcp.org/en/jst/detail?id=177>
- [74] Y. Oren, A.-R. Sadeghi, and C. Wachsmann, "On the effectiveness of the remanence decay side-channel to clone memory-based PUFs," in *Proc. 15th Int. Conf. Cryptogr. Hardware Embedded Syst.*, 2013, pp. 107–125.
- [75] B. Parno, J. M. McCune, and A. Perrig, "Bootstrapping trust in commodity computers," in *Proc. IEEE Symp. Security Privacy*, 2010, pp. 414–429.
- [76] N. L. Petroni, Jr., T. Fraser, J. Molina, and W. A. Arbaugh, "Copilot—A coprocessor-based kernel runtime integrity monitor," in *Proc. 13th Conf. USENIX Security Symp.*, 2004, vol. 13.
- [77] B. Pfizmann, J. Riordan, C. Stübke, M. Waidner, and A. Weber, "The PERSEUS system architecture," IBM Res., Tech. Rep. RZ 3335 (#93381), 2001.
- [78] D. C. Ranasinghe, D. W. Engels, and P. H. Cole, "Security and privacy: Modest proposals for low-cost RFID systems," 2004.
- [79] RSA Laboratories, "PKCS#11 v2.20: Cryptographic token interface standard," 2004. [Online]. Available: <http://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20.pdf>
- [80] U. Rührmair et al., "Modeling attacks on physical unclonable functions," in *Proc. 17th ACM Conf. Comput. Commun. Security*, 2010, pp. 237–249.
- [81] U. Rührmair and M. van Dijk, "PUFs in security protocols: Attack models and security evaluations," in *Proc. IEEE Symp. Security Privacy*, 2013, pp. 286–300.
- [82] A.-R. Sadeghi and D. Naccache, Eds., *Towards Hardware-Intrinsic Security*. New York, NY, USA: Springer-Verlag, 2010.
- [83] A.-R. Sadeghi and C. Stübke, "Property-based attestation for computing platforms: Caring about properties, not mechanisms," in *Proc. New Security Paradigms Workshop*, 2004, pp. 67–77.
- [84] A.-R. Sadeghi, I. Visconti, and C. Wachsmann, "Enhancing RFID security and privacy by physically unclonable functions," in *Towards Hardware-Intrinsic Security*. New York, NY, USA: Springer-Verlag, 2010.
- [85] R. Sailer et al., "Building a MAC-based security architecture for the Xen open-source hypervisor," in *Proc. 21st Annu. Comput. Security Appl. Conf.*, 2005, DOI: 10.1109/CSAC.2005.13.
- [86] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn, "Design and implementation of a TCG-based integrity measurement architecture," in *Proc. 13th Conf. USENIX Security Symp.*, 2004, vol. 13.
- [87] D. Schellekens, B. Wyseur, and B. Preneel, "Remote attestation on legacy operating systems with trusted platform modules," *Sci. Comput. Program.*, vol. 74, no. 1–2, pp. 13–22, 2008.
- [88] S. Schulz and A.-R. Sadeghi, "Secure VPNs for trusted computing environments," in *Proc. 2nd Int. Conf. Trusted Comput.*, 2009, pp. 197–216.
- [89] S. Schulz, A.-R. Sadeghi, and C. Wachsmann, "Lightweight remote attestation using physical functions," *Cntr. Adv. Security Res. Darmstadt (CASED)*, Darmstadt, Germany, Tech. Rep., 2011.
- [90] S. Schulz, A.-R. Sadeghi, and C. Wachsmann, "Short paper: Lightweight remote attestation using physical functions," in *Proc. 4th ACM Conf. Wireless Netw. Security*, 2011, pp. 109–114.
- [91] M. Selhorst and C. Stübke, "TrustedGRUB Project," 2010. [Online]. Available: <http://sf.net/projects/trustedgrub/>
- [92] A. Seshadri et al., "Pioneer: Verifying integrity and guaranteeing execution of code on legacy platforms," in *Proc. 20th Symp. Oper. Syst. Principles*, 2005, DOI: 10.1145/1095810.1095812.
- [93] A. Seshadri, A. Perrig, L. van Doorn, and P. K. Khosla, "SWATT: SoftWare-based ATTestation for embedded devices," in *Proc. IEEE Symp. Security Privacy*, 2004, pp. 272–282.
- [94] M. Shaneck, K. Mahadevan, V. Kher, and Y. Kim, "Remote software-based attestation for wireless sensors," in *Security and Privacy in Ad-Hoc and Sensor Networks*. New York, NY, USA: Springer-Verlag, 2005.
- [95] S. W. Smith, "Outbound authentication for programmable secure coprocessors," in *Proc. 7th Eur. Symp. Res. Comput. Security*, 2002, pp. 72–89.
- [96] J. Srage and J. Azema, "M-shield mobile security technology, T1 white paper," 2005. [Online]. Available: [http://focus.ti.com/pdfs/wtbu/ti\\_mshield\\_whitepaper.pdf](http://focus.ti.com/pdfs/wtbu/ti_mshield_whitepaper.pdf)
- [97] R. Strackx and F. Piessens, "Fides: Selectively hardening software application components against kernel-level or process-level malware," in *Proc. ACM Conf. Comput. Commun. Security*, 2012, pp. 2–13.
- [98] R. Strackx, F. Piessens, and B. Preneel, "Efficient isolation of trusted subsystems in embedded systems," in *Security and Privacy in Communication Networks (SecureComm)*. New York, NY, USA: Springer-Verlag, 2010.
- [99] E. Suh, C. O'Donnell, I. Sachdev, and S. Devadas, "Design and implementation of the AEGIS single-chip secure processor using physical random function," in *Proc. 32nd Int. Symp. Comput. Architect.*, 2005, pp. 25–36.
- [100] S. Tamrakar, "Tapping and tripping with NFC," *Trust and Trustworthy Computing (TRUST)*, vol. 7904. Berlin, Germany: Springer-Verlag, 2013, pp. 115–132, ser. Lecture Notes in Computer Science.
- [101] Trusted Computing Group. [Online]. Available: <https://www.trustedcomputinggroup.org/>
- [102] Trusted Platform Module (TPM) Specifications. [Online]. Available: <https://www.trustedcomputinggroup.org/specs/TPM/>
- [103] V. van der Leest, G.-J. Schrijen, H. Handschuh, and P. Tuyls, "Hardware intrinsic security from D flip-flops," in *Proc. 5th ACM Workshop Scalable Trusted Comput.*, 2010, pp. 53–62.
- [104] A. Vasudevan, J. McCune, J. Newsome, A. Perrig, and L. V. Doorn, "CARMA: A hardware tamper-resistant isolated execution environment on commodity x86 platforms," in *Proc. 7th ACM Symp. Inf. Comput. Commun. Security*, 2012, pp. 48–49.
- [105] Verayo Inc., "Products," 2013. [Online]. Available: <http://www.verayo.com/product/products.html>
- [106] M. Wilkes and R. Needham, *The Cambridge CAP Computer and Its Operating System*. Amsterdam, The Netherlands: Elsevier, 1979.
- [107] W. Wulf et al., "HYDRA: The kernel of a multiprocessor operating system," *Commun. ACM*, vol. 17, no. 6, pp. 337–345, 1974.
- [108] B. S. Yee, "Using secure coprocessors," Ph.D. dissertation, Schl. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, 1994, CMU-CS-94-149.
- [109] "Bundesbehörden sehen Risiken beim Einsatz von Windows 8," *Zeit Online*, 2013. [Online]. Available: <http://www.zeit.de/digital/datenschutz/2013-08/trusted-computing-microsoft-windows-8-nsa>
- [110] Z. Zhou, V. D. Gligor, J. Newsome, and J. M. McCune, "Building verifiable trusted path on commodity x86 computers," in *Proc. Symp. Security Privacy*, 2012, pp. 616–630.

## ABOUT THE AUTHORS

**N. Asokan** received the Ph.D. degree in computer science from the University of Waterloo, Waterloo, ON, Canada.

Currently, he is a Professor of Computer Science and Engineering at Aalto University, Aalto, Finland and a Professor of Computer Science at the University of Helsinki, Helsinki, Finland. He is the lead academic principal investigator of the Intel Collaborative Research Institute for Secure Computing in Finland. Prior to joining academia, he spent several years in industrial research, first with IBM Research and then with Nokia Research Center. He is interested in understanding how to build systems that are simultaneously secure, easy to use and inexpensive to deploy.



**Jan-Erik Ekberg** received the Ph.D. degree in computer science from Aalto University, Aalto, Finland, and holds a Licentiate degree in electrical engineering from the same university.

Currently, he is the Director of Advanced Development at Trustonic, Aalto, Finland. His background is in research and the telecom industry, where he worked for 18 years in various research positions. His primary interests are with issues related to mobile platform security and trusted execution environments, but he has also background in securing network protocols and telecom systems, as well as with short-range communication technologies such as NFC, BT-LE, and WLAN. He has participated in several standardization activities (TCG/MPWG, BTSIG/ BTLE), and holds 30+ patents.



**Kari Kostinen** received the Ph.D. degree in computer science from Aalto University, Aalto, Finland. In his dissertation he studied mobile platform security mechanisms and took part in developing onboard credentials platform. His dissertation results have been included in both deployed products and industry standards.

Currently, he is a Postdoctoral Researcher at System Security Group of ETH Zurich, Zurich, Switzerland. His research focus is on security and privacy issues of mobile devices. Before joining ETH, he spent several years at Nokia Research Center, Helsinki, Finland and also briefly in Palo Alto, CA, USA. His current research topics range from secure smartphone applications to hardware security mechanisms.



**Anand Rajan** is the Director of the Emerging Security Lab at Intel Labs, Hillsboro, OR, USA. He leads a team of senior technologists whose mission is to research novel security features that raise the assurance of platforms across the compute continuum (cloud to wearables). The topics covered by his team span trustworthy execution environments, mobile security, identity & authentication, cryptography, and security for emerging paradigms (IOT, wearables). He has been a Principal Investigator for Intel's research collaboration with academia, government, and commercial labs in the area of trustworthy platforms. He and his team developed the Common Data Security Architecture specification, adopted as worldwide standard by The Open Group. His team was also instrumental in several security standardization efforts (e.g., PKCS#11, BioAPI, UPnP-Security, & EPID). Prior to joining Intel in



1994, he was the Technical Lead for the Trusted-UNIX team at Sequent Computer Systems and worked on development and certification of a TCSEC B1-level Operating System.

Dr. Rajan was an active member of the IEEE WG that crafted the P1363 (public-key crypto) standard.

**Carlos Rozas** received the B.S. and M.S. degrees in computer engineering from the University of Michigan, Ann Arbor, MI, USA.

Currently, he is a Principal Engineer in Intel Labs, Hillsboro, OR, USA, where he leads research efforts in cloud security and is Intel Software Guard Extensions architect. In the past, he has led research in combining trustworthy computing and virtualization. This included the development of TPM virtualization technology and the development of runtime software verification technologies. He has worked on software tamper resistance and verification technologies used by the Common Data Security Architecture. He has published seven technical papers and holds 23 patents.

Mr. Rozas received an Intel Achievement Award for his work on Intel Software Guard Extensions.



**Ahmad-Reza Sadeghi** received the Ph.D. degree in computer science with the focus on privacy protecting cryptographic protocols and systems from the University of Saarland, Saarbrücken, Germany.

Currently, he is the Head of the System Security Lab at the Center for Advanced Security Research Darmstadt (CASED), Technische Universität Darmstadt, Darmstadt, Germany and the Scientific Director of Fraunhofer Institute for Secure Information Systems (SIT). Since January 2012, he has been the Director of Intel-TU Darmstadt Security Institute for Mobile and Embedded Systems in Darmstadt, Germany. Prior to academia, he worked in Research and Development of Telecommunications enterprises, among others Ericson Telecommunications. He has led and been involved in a variety of national and international research and development projects on design and implementation of trustworthy computing platforms and trusted computing, security hardware, physically unclonable functions (PUFs), cryptographic privacy-protecting systems, and cryptographic compilers (in particular for secure computation).

Prof. Sadeghi has been continuously contributing to the IT security research community and serving as General Chair or Program Chair as well as Program Committee Member of many conferences and workshops in information security and privacy, trusted computing, and applied cryptography. He is on the Editorial Board of the *ACM Transactions on Information and System Security*.



**Steffen Schulz** received the Ph.D. degree, researching trusted channels and trust anchors in low-cost embedded systems at Macquarie University, Sydney, N.S.W., Australia and Ruhr-University Bochum, Bochum, Germany.

Currently, he is a Security Researcher at Intel Labs and is currently situated at the Intel Collaborative Research Institute for Secure Computing (ICRI-SC), Darmstadt, Germany. He was involved in multiple national and international research projects and coauthored several publications in the area of network security, lightweight/scalable attestation, and trusted execution.



**Christian Wachsmann** received the Ph.D. degree in computer science from the Technische Universität Darmstadt, Darmstadt, Germany.

Currently, he is a Postdoctoral Researcher at the Intel Collaborative Research Institute for Secure Computing (IC-RISC), Technische Universität Darmstadt. His current research focuses on the design, development, formal modeling, and security analysis of security architectures and cryptographic protocols to verify the software integrity (attestation) of embedded systems. He is the main coauthor of more than 30 scientific articles in internationally renowned journals and conferences on IT security.



# Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain

*This paper surveys the state of the art in counterfeiting and detection technologies.*

By UJJWAL GUIN, *Student Member IEEE*, KE HUANG, *Member IEEE*, DANIEL DiMASE,  
JOHN M. CARULLI, JR., *Senior Member IEEE*, MOHAMMAD TEHRANIPOOR, *Senior Member IEEE*, AND  
YIORGOS MAKRIS, *Senior Member IEEE*

**ABSTRACT** | As the electronic component supply chain grows more complex due to globalization, with parts coming from a diverse set of suppliers, counterfeit electronics have become a major challenge that calls for immediate solutions. Currently, there are a few standards and programs available that address the testing for such counterfeit parts. However, not enough research has yet addressed the detection and avoidance of all counterfeit parts—recycled, remarked, overproduced, cloned, out-of-spec/defective, and forged documentation—currently infiltrating the electronic component supply chain. Even if they work initially, all these parts may have reduced lifetime and pose reliability risks. In this tutorial, we will provide a review of some of the existing counterfeit detection and avoidance methods. We will also discuss the challenges ahead for implementing these methods, as well as the development of new detection and avoidance mechanisms.

**KEYWORDS** | AC/DC parametric tests; counterfeit integrated circuits (ICs); electrical inspection; hardware security; machine learning; path-delay test; physical inspection

## I. INTRODUCTION

Counterfeit integrated circuits (ICs), a major source of concern in the electronic component supply chain because of reliability and security issues, are impacting many industrial sectors, including computers, telecommunications, automotive electronics, and even military systems [1], [2]. The consequences can, obviously, be dramatic when critical systems begin to fail due to the use of counterfeit or low-quality components. According to [3], legitimate electronics companies miss out on about \$100 billion of global revenue every year because of counterfeiting. Indeed, the hi-tech industry is significantly impacted by counterfeiting activity. Around 1% of semiconductor sales are estimated to be those of counterfeited units [4]. The tools and technologies utilized by counterfeiters have become extremely sophisticated and well financed [5]. In turn, this also calls for more sophisticated methods to detect counterfeit electronic parts that enter the market. Data provided by IHS (Englewood, CO, USA), shown in Fig. 1, shows that reports of counterfeit parts have quadrupled since 2009.

Counterfeit ICs pose a significant threat to the global electronics component supply chain and are becoming more difficult to detect as the counterfeiters increase their level of sophistication [7]. Counterfeiters are improving their technique and expertise, to an extent of successfully

Manuscript received October 16, 2013; revised April 22, 2014; accepted June 9, 2014.  
Date of publication July 15, 2014; date of current version July 18, 2014.

**U. Guin** and **M. Tehranipoor** are with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269 USA (e-mail: [ujjwal@engr.uconn.edu](mailto:ujjwal@engr.uconn.edu); [tehrani@engr.uconn.edu](mailto:tehrani@engr.uconn.edu)).

**K. Huang** is with the Department of Electrical and Computer Engineering, San Diego State University, San Diego, CA 92115 USA (e-mail: [k.h.huang@ieee.org](mailto:k.h.huang@ieee.org)).

**D. DiMase** is with Honeywell Inc., Providence, RI 02895 USA (e-mail: [Daniel.DiMase@Honeywell.com](mailto:Daniel.DiMase@Honeywell.com)).

**J. M. Carulli, Jr.** is with Texas Instruments, Dallas, TX 75243 USA (e-mail: [j.m.carulli@ieee.org](mailto:j.m.carulli@ieee.org)).

**Y. Makris** is with the Department of Electrical Engineering, The University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: [yiorgos.makris@utdallas.edu](mailto:yiorgos.makris@utdallas.edu)).

Digital Object Identifier: 10.1109/JPROC.2014.2332291

0018-9219 © 2014 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See [http://www.ieee.org/publications\\_standards/publications/rights/index.html](http://www.ieee.org/publications_standards/publications/rights/index.html) for more information.

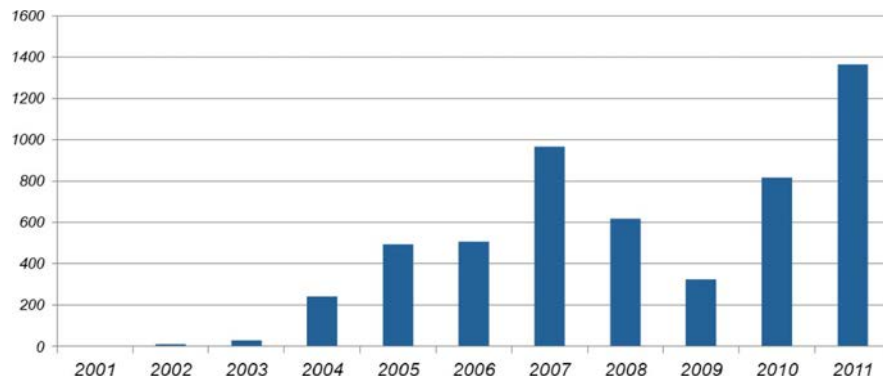


**Fig. 1.** Reported counterfeit parts have been quadrupled since 2009 [6].

duplicating a company itself. Counterfeit ICs are of great concern to industry and government because a system malfunction can present situations that cause mission failures and health or safety concerns [8]. The potential for loss and tragedy caused by such devices could be significant for electronic systems supporting a number of sectors (e.g., medical, aerospace, defense, automotive, banking, energy/smart-grid, etc.).

Table 1 shows the top-5 most counterfeited semiconductors in 2011 according to data from ERAI (Naples, FL, USA). ERAI is a private organization that collects reported data on counterfeit incidents and reports the information to their members. As the table shows, the counterfeits with the most reported incidents impact affect both analog and digital ICs, as well as discrete components.

Contemporary advancements in very large scale integration (VLSI) have been accompanied by increasing variation in the performances of fabricated chips and concerns about the correctness of their operation. Indeed, failures can occur at any stage in the lifetime of an IC. During production, devices can fail due to design weaknesses, excessive process variations, or local spot defects. After production, devices can fail due to defects which are not detected by the production tests and manifest themselves later in the field of operation. These early life failures are caused by extrinsic process defects and are known as infant mortality. ICs can also fail during their lifetime due to aging, wear-and-tear, harsh environments, overuse, etc. These failures occur when a material or component exceeds its fundamental capability and are known as intrinsic reliability failure mechanisms.

**Table 1** Top-5 Most Counterfeited Semiconductors in 2011 [10]

| Ranks | Component type        | % of reported incidents |
|-------|-----------------------|-------------------------|
| 1     | Analog IC             | 25.2%                   |
| 2     | Microprocessor IC     | 13.4%                   |
| 3     | Memory IC             | 13.1%                   |
| 4     | Programmable logic IC | 8.3%                    |
| 5     | Transistor            | 7.6%                    |

Depending on the end-user application, ICs may go through burn-in tests, where they are exercised sufficiently long under stress conditions, in order to avoid early in-use system failures or to estimate the operating life of a particular device. Once the reliability issues of an IC are properly addressed, its lifetime can be estimated, and it can be shipped to customers. Fig. 2 illustrates typical device failure characteristics, often known as the bathtub curve [11]. The failure rate is defined as the probability that a device will fail in the time interval between  $t$  and  $t + \delta t$ , given that it has survived until time  $t$  [12]. As can be observed in Fig. 2, counterfeit devices are expected to have shorter time to failure compared to brand new devices. Table 2 summarizes the possible effects of counterfeit ICs for governments, industries, and consumers.



**Fig. 2.** The classical bathtub curve illustrating typical device failure characteristics.

**Table 2** Possible Effects of Counterfeit ICs [13]

| Government                                                 | Industry                                     | Consumer                                                                           |
|------------------------------------------------------------|----------------------------------------------|------------------------------------------------------------------------------------|
| National security or civilian safety issues                | Costs to mitigate this risk                  | Costs when products fail due to lower quality and reliability of counterfeit parts |
| Costs of enforcement                                       | Costs to replace failed parts                |                                                                                    |
| Lost tax revenue due to illegal sales of counterfeit parts | Lost sales                                   | Potential safety concerns                                                          |
|                                                            | Lost brand value or damage to business image |                                                                                    |

As the IC supply chain has become globalized and thus more complex, additional sources of failure have become a concern. Specifically, the trustworthiness of IC supply chain is much harder to assess. Indeed, ICs provided by untrusted points in the supply chain could be intentionally relabeled, illegitimately replicated, or recycled from used or defective circuit boards. Even if these ICs work initially, they may have a reduced lifetime and can pose reliability risks.

### A. Survey of Articles

It was pointed out by the Semiconductor Industry Association (SIA, Washington, DC, USA) in [8] that counterfeit devices can be avoided by exclusively buying semiconductor products either directly from the original component manufacturer (OCM) or directly from the OCM-authorized distributors/resellers. However, it is also reported by the SAE International (Washington, DC, USA) and the U.S. Department of Commerce that authorized distributors encounter counterfeit parts [2]. The problem is further compounded due to obsolescence and life cycles of piece parts which have much shorter life cycles than complex systems which the parts are incorporated within, requiring significant requalification. Several practices have been developed to identify counterfeit devices. They are broadly classified into counterfeit detection and counterfeit avoidance methods. For the purposes of this paper, counterfeit detection focuses on the detection of counterfeit parts in the supply chain whereas counterfeit avoidance concentrates on adding extra hardware in the circuit such that a suspect part is authenticated without costly and time-consuming detection methods (e.g., design for counterfeit avoidance, design for test, design for security).

In [14] and [15], Guin *et al.* 1) developed a comprehensive taxonomy of counterfeit types, counterfeit detection methods, and counterfeit defects; 2) presented assessment of all currently available counterfeit detection methods; and 3) proposed a method selection technique to maximize the test coverage specific to a set of tests. In [16], a statistical approach is used to distinguish recycled counterfeit ICs by training a one-class classifier using only new devices. The measurements used to build the classifier are typical test results from production early failure rate (EFR) analysis required to release most products, such as  $V_{\min}$ ,  $F_{\max}$ , and  $I_{ddq}$ , thus no additional costs in terms of design, test, and area/power overhead are incurred to perform identification. The method is demonstrated using measurements from new and aged devices taken from actual chips in production. Path-delay fingerprinting, a method first introduced in [17] in the context of hardware Trojan detection, is adapted in [18] for detecting recycled ICs. This method also assumes that the recycled ICs have aged due to usage in the field, thus their path-delay distribution changes, providing opportunity for detection.

The most well-known counterfeit avoidance techniques include secure split test (SST), hardware metering, physical unclonable functions (PUFs), lightweight on-chip sensors, package ID, etc. SST attempts to prevent overproduced, out-

of-spec/defective, and cloned ICs from entering into the supply chain. It enables the design house back into the manufacturing test process by placing a set of security measures in the design and controlling the test flow [19]. Hardware metering attempts to uniquely tag each chip produced from a certain design by active or passive methods to facilitate chip tracing [20], [21]. Similarly, part authentication tools [22] consist of providing an encrypted number for each device by a radio-frequency identification (RFID) tag in production. However, reverse-engineering tools have become very advanced and allow an attacker to read the stored encrypted number. To overcome this challenge, hardware intrinsic security (HIS) has been proposed as a mechanism that can provide security based on the inherent properties of an electronic device [23]. PUFs [24], for example, belong to the category of such HIS mechanisms. PUFs aim to measure the responses of hardware to certain given inputs, which depend on the unique physical properties of the device, since process variations affect each device in a unique and unclonable fashion. On-chip aging sensors and chip usage measurement structures have been proposed in an effort to detect recycled counterfeit devices [25]–[27]. Finally, package IDs have been proposed to track the components in the supply chain [28], [29].

### B. Taxonomy of Counterfeit ICs

A counterfeit component 1) is an unauthorized copy; 2) does not conform to OCM design, model, and/or performance standards; 3) is not produced by the OCM or is produced by unauthorized contractors; 4) is an off-specification, defective, or used OCM product sold as “new” or working; or 5) has incorrect or false markings and/or documentation [2]. Based on the definition above and analyzing supply chain vulnerabilities, we classify the counterfeit types into seven distinct categories [7], [15], [30].

Fig. 3 shows the taxonomy of such counterfeit components. Recycled and remarked components draw much attention in the media, test labs, and industry, and they jointly contribute more than 80% of counterfeit incidents [31]. The recycled components are taken from used printed circuit boards (PCBs), repackaged and remarked, and then sold in the market as new. The remarking process includes the removal of markings on the package (or even on the die) and remarking with forged information. New components can also be remarked to obtain a higher specification, such as remarking from commercial grade part to industrial or defense grade. In overproduction, unauthorized actors in a foundry, assembly, or test site that have access to a designer’s IP also have the ability to fabricate ICs outside contract. They can then sell excess ICs in the open market. The unauthorized actor may either knowingly sell out-of-spec/defective components, or they may be stolen and sold on open markets. Cloning is a process of copying a design by counterfeiters mainly to reduce the large development cost of a component. Cloning can be done in two ways: by reverse engineering,

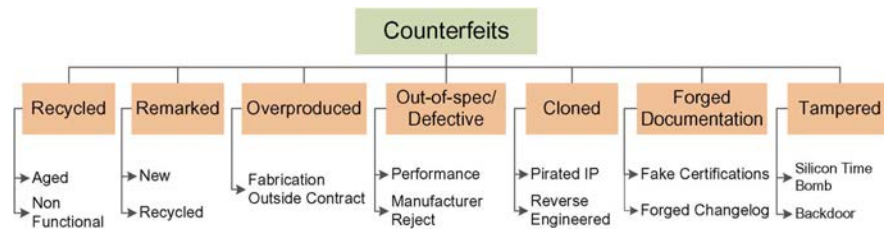


**Fig. 3. Taxonomy of counterfeit types [7], [14], [30].**

and, by obtaining IPs illegally. Forged documentation may include the false certification of compliance for some standards or programs, or a falsified revision history or change log of a component. The final category of counterfeits is the tampered type. Tampering can occur at the die level (“hardware Trojan”) or package level or in the software or firmware of the device. Such components can either act as a silicon time bomb where the device can behave differently under certain conditions or act as a backdoor where secret information from the chip can be sent out to an adversary [32].

### C. Standardization Activities

Counterfeit prevention and detection require global recognition of the problem and a unified solution. Such a solution often comes in the form of international standards and may be verified through accredited conformity assessment programs. These programs may include key elements such as technical standard(s) recognized worldwide, competency-based training program(s), global supply chain certification system(s), certification bodies, and accreditation schemes. There are standards and command media in place or in development that includes guidance or requirements for detection of the counterfeit parts. One committee responsible for many of these standards is the G-19 Counterfeit Electronic Parts Committee, set forth by SAE International. Their standards target different sectors of the industry: independent distributors and brokers, authorized distributors, users and integrators (including government agencies), and test service providers. These standards are as follows:

- 1) AS5553: Counterfeit Electronic Parts; Avoidance, Detection, Mitigation, and Disposition [33];
- 2) ARP6178: Fraudulent/Counterfeit Electronic Parts; Tool for Risk Assessment of Distributors [34];
- 3) AS6081: Fraudulent/Counterfeit Electronic Parts: Avoidance, Detection, Mitigation, and Disposition—Distributors Counterfeit Electronic Parts; Avoidance Protocol, Distributors [35] (intended for independent distributors and brokers);
- 4) AS6496: Fraudulent/Counterfeit Electronic Parts: Avoidance, Detection, Mitigation, and Disposition—Authorized/Franchised Distribution [36];

- 5) AS6171: Test Methods Standard; Counterfeit Electronic Parts [37].

Components Technology Institute, Inc. (CTI, Huntsville, AZ, USA) has also created a Counterfeit Components Avoidance Program (CCAP-101) [38]. Independent distributors can be certified as CCAP-101 compliant, which is done by means of a yearly audit. Independent Distributors of Electronics Association (IDEA, West Baden Springs, IN, USA) has developed IDEA-STD-1010-B which mainly provides guidance for the visual inspection of electronic components [39].

Currently, the major issue with many of these standards is that they address the parts that are already circulating in the market, mainly, recycled and remarked parts. Any of the current counterfeit detection standards are not capable of addressing the detection of all types of counterfeits. Moreover, none of these standards are intended for developing avoidance mechanisms in ICs.

### D. Taxonomy of Component Types

The type of components can significantly impact detection and avoidance. Components can be classified into three distinct types, namely, obsolete, active, and new. Components become obsolete when the OCM stops manufacturing them. The OCM may produce newer designs and no longer sell the previous generation. They may only be available through the electronic components distributors. Active parts are those that companies continue to fabricate, however, the design is congealed. In these cases, there may be a possibility of modifying the package instead of the die design of a component to address anti-counterfeit measures. New components are very flexible to implement any anti-counterfeit measures. The OCM can decide if one or more of these measures could be placed in the design depending on the area, power, and cost constraints.

## II. COUNTERFEIT DETECTION

Over the past several years a specialized service of testing has been created for detecting counterfeit components. The components must be authenticated by these tests before being placed in systems. Fig. 4 shows a generalized taxonomy of counterfeit detection methods. The methods are broadly classified into physical and electrical inspections.

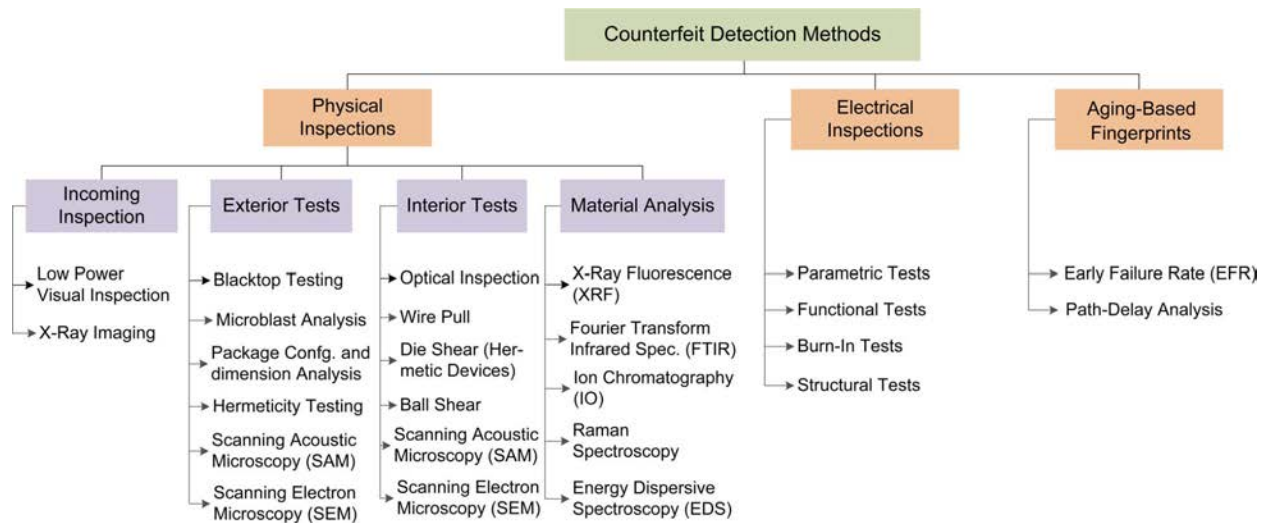


Fig. 4. Taxonomy of counterfeit detection methods [7], [14], [15].

### A. Physical Inspection for Counterfeit Detection

Such inspections are based on the physical properties of the component. These tests are grouped into four categories. In incoming inspection, all the components are inspected thoroughly. The external structure is observed carefully by low-power visual inspection (LPVI) while the internal structure is inspected by X-ray imaging. There are several external tests, such as blacktop testing, microblast analysis, hermeticity testing, scanning electron microscopy (SEM), scanning acoustic microscopy (SAM), and a variety of other test methods recommended to find the defects and anomalies present outside the package, inside the package, and on the leads of a component. For interior tests, one needs to decap the component first to expose the internal structure. Optical inspection, wire pull, die/ball shear, and SEM are a few of the tests that test labs generally perform for internal inspection. Material analysis methods are performed to find the defects related to the material composition of the package, leads, and die. X-ray fluorescence (XRF), Fourier transform IR spectroscopy (FTIR), and energy dispersive spectroscopy (EDS), are a few material analysis methods.

Physical inspections are usually the first set of tests to be performed on the incoming components for authentication. The methodology and implementation of these tests apply uniformly on all types of components—obsolete, active, and new. These tests are based on the physical properties of leads, package, and die of components. In this section, we will describe some commonly used physical inspections used for the detection of counterfeit components.

1) *Low-Power Visual Inspection (LPVI)*: LPVI is the first test usually performed on all the components. The leads and packages are carefully examined using a low-power microscope or magnification lamp, generally with less than 10X magnification. All the relevant information, packaging and

shipping information, part number, lot/date/country code, etc., is documented in detail. Recycled parts, desoldered from the PCB, can sometimes be observed with deformed leads and extra material on them. Sometimes, a residual trace of the original marking exists below the new one. Scratches are often visible on the package as a sign of recycling.

Fig. 5 shows four counterfeit defects and anomalies detected by LPVI. Fig. 5(a) displays a peeled off plating layer from the leads. The leads in Fig. 5(b) clearly show a possible rework or reflow soldering. Ghost marking (residual marking

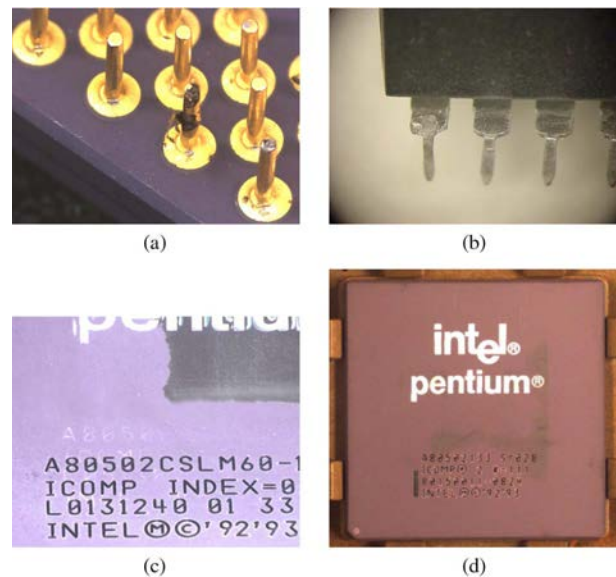


Fig. 5. Counterfeit defects detected by LPVI (source: Honeywell). (a) Fake plating on leads. (b) Residual materials on leads. (c) Ghost marking on the package. (d) Heat sink mark on the package.

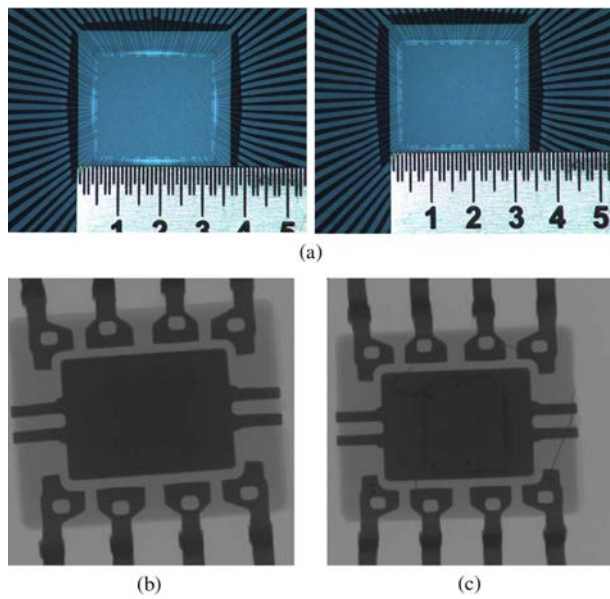


**Fig. 6. Counterfeit defects detected by X-ray imaging**  
(source: Honeywell). (a) Wrong Die. (b) Missing bond wires.  
(c) Broken bond wires.

in white color below the new one) is clearly visible in Fig. 5(c). The heat sink mark in Fig. 5(d) indicates the prior usage of this Intel chip. These defects are easy to detect with LPVI.

2) *X-Ray Imaging*: X-ray imaging is a method of inspecting the internal structure of a component without performing the decapsulation. It generally belongs to the nondestructive category of tests. There are typically two types of X-ray imaging systems: film X-ray and real-time X-ray systems. In film X-ray systems, the images are formed on a radiographic film, whereas in real-time X-ray systems, a digital image is formed by digital sensors. The defects and anomalies related to die and bond wires such as missing or wrong die, cracks on the die, broken bond wires, etc., may be detected. Additional tests may be required to complete the authenticity of a component.

X-ray imaging is an important method used for counterfeit detection. It is performed on components to verify that the internal package, bond wires, and die construction are consistent with a golden (reference) part. If the golden part is not available, comparisons should be carried out within the same lot. Fig. 6 shows some typical counterfeit defects detected by X-ray imaging. Fig. 6(a) shows two different die sizes in the same lot. There are no bond wires inside the package that is shown in Fig. 6(b). Broken bond wires are presented in Fig. 6(c).

3) *Microblast Analysis*: Microblasting is a dry blasting counterfeit technique in which accurately defined blasting agents are bombarded on the surface of the target device in an effort to remove part markings and scratches from

recycled and used parts or falsely represent new parts (e.g. upgrading temperature or speed-grade) by removing part markings prior to Counterfeit Electronic Parts remarking devices. Compressed air is generally used to accelerate the blasting particles. Some popular blasting agents, such as aluminum oxide powder, sodium bicarbonate powder, and glass bead, are used, depending on the components' package type [such as dual in-line package (DIP), plastic leaded chip carrier (PLCC), etc.]. A variety of surface analysis techniques may be deployed to detect microblasting. Test methods include SEM, FTIR, RAMAN, and high-power microscopy, at typically more than 200x or greater.

4) *Scanning Acoustic Microscopy (SAM)*: SAM is one of the efficient ways of studying the structure of a component without damaging it. This technology functions by utilizing the reflection and transmission of ultrasound waves to generate an image of the component based on its acoustic impedance at various depths. The component under test is submerged in either deionized water or isopropyl alcohol (IPA), which is used as a medium. Since air will have a much different acoustic impedance than any of the part's mediums, that section will appear much darker on the image produced. The resolution of SAM depends on the transducer frequencies. Lower frequencies provide higher penetration through the component at the cost of lower spatial resolution. SAM is very useful in detecting delamination, or, die attachment to the package. It can also detect the cracks and voids in the die and anomalies in the bond wires. SAM can also be useful for surface analysis and detecting ghost markings or sanding marks using reflective mode.

5) *Scanning Electron Microscopy (SEM)*: SEM is a method of generating an image with a superfine resolution by using a focused electron beam. The image is formed by scanning the entire target area of the sample. SEM consists of two major components: the electron column and a control console. The column generates the focused electron beam for scanning the surface and the control console displays the image. When the high energetic electron beam interacts with the sample, it generates a secondary emission of backscattered electrons and X-rays. An electron detector detects these secondary electrons and an image is formed. A detailed description can be found in [40]. SEM is very useful for detecting many defects and anomalies present in counterfeit components. Using SEM to inspect the die requires decapsulation of the component. However, for external inspections, it is not necessary to decapsulate. The major issue associated with SEM is the long test time. Sometimes it requires several hours to inspect a single component in detail.

6) *X-Ray Fluorescence (XRF) Spectroscopy*: XRF is a nondestructive method for material analysis. The emission characteristics of a material are observed after heavy bombardment of high-energy X-rays. When the X-ray hits the surface of a material, the outer electrons obtain enough energy

**Table 3** XRF Measurement Results (Source: Integra Technologies)

| mil SnPb |        | % Sn |         | % Pb |        |
|----------|--------|------|---------|------|--------|
| Row      | Sample | Row  | Sample  | Row  | Sample |
| 1        | 0.193  | 1    | 100.000 | 1    | 0.000  |
| 2        | 0.243  | 2    | 100.000 | 2    | 0.000  |
| 3        | 0.351  | 3    | 100.000 | 3    | 0.000  |
| 4        | 3.495  | 4    | 99.876  | 4    | 0.124  |
| 5        | 0.853  | 5    | 99.829  | 5    | 0.171  |
| 6        | 0.030  | 6    | 100.000 | 6    | 0.000  |

\* Part Number: TAJD476K020R and Pin Package: 2 CHIPCAP.

(ionization potential) to reach unstable higher outer orbits. The emission of radiation occurs when these high-energy electrons settle down to their original ground state. Each element produces a unique peak in the spectrum. A unique fingerprint is generated from the package of a component by XRF spectroscopy. A decision about a component's authenticity can be made upon comparison with a golden sample or the manufacturer data sheet if available. There are several X-ray fluorescence spectrometers with an automated sample feed that are available for material analysis. Table 3 shows the XRF measurement results. Rows 1–3 and 4–6 represent the known good and suspect samples, respectively. The suspect sample was a board pull, and shows higher lead coating thickness, and also some lead content in it.

7) *Fourier Transform Infrared (FTIR) Spectroscopy*: FTIR works based on infrared (IR) spectroscopy. A part of IR radiation is absorbed by the material under test and the other part is transmitted through it. The spectrum for molecular absorption and transmission is observed from the resultant IR radiation. The unique molecular fingerprint, created by FTIR, is compared to the fingerprint of the golden model for material comparison. FTIR is used to authenticate both organic and inorganic materials of a component. It is used to verify: 1) polymer, coating, etc., of the package; 2) residual foreign materials from the sand blasting process used to remove the old markings; and 3) residuals from chemical processing typical from counterfeits performing part removal from printed circuit boards and from the illicit refurbishment process.

8) *Energy Dispersive Spectroscopy (EDS)*: EDS is used to chemically characterize a component using X-ray excita-

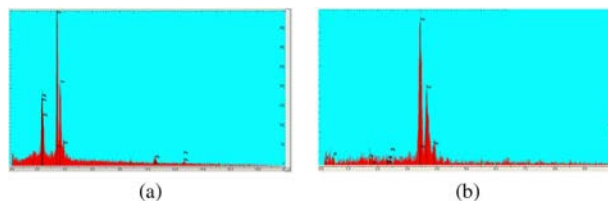


**Fig. 7.** Counterfeit defects detected by EDS (source: Honeywell).  
**(a)** Counterfeit: element lead found in the leads of an IC.  
**(b)** Genuine: No lead found.

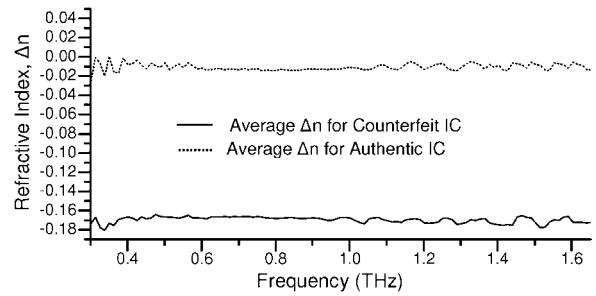


**Fig. 8.** Difference in refractive index between top and bottom sides of ICs as a function of radiation frequency [42].

tion. A high-energy beam of charged particles is bombarded on the surface, and the emitted X-ray spectrum is captured by an X-ray detector to form the EDS spectrum. A unique fingerprint of X-ray spectrum is generated by the materials used in the component's packaging. Fig. 7 shows the EDS spectrum generated from the leads of an IC. The material lead is detected on the leads of a counterfeit component during lead finish testing, shown in Fig. 7(a). The genuine component does not have the material lead on leads of the IC [Fig. 7(b)].

9) *Terahertz Time Domain Spectroscopy (THz-TDS)*: THz imaging [41] is used to inspect the internal structure of a component by using a pulsed laser operated in the THz frequency range. This technique does not require the application of any bias to the component during testing. There is also no need to decapsulate the component to observe the internal structure. Differential refractive indexes of component die and packaging allows the identification of counterfeit electronics [42]. THz spans the frequency region between 100 GHz and 30 THz. Three attributes make THz imaging useful for counterfeit detection: THz is 1) fully absorbed by metal; 2) partially absorbed by a doped semiconductor; and 3) transparent to plastics. Fig. 8 demonstrates that a counterfeit IC has a completely different refractive index than the a genuine one. The x-axis and the y-axis of the figure represent the frequency of the pulsed laser and differential refractive index ( $\Delta n$ ), respectively. A large  $\Delta n$  has been observed in the counterfeit IC.

## B. Electrical Inspection for Counterfeit Detection

These inspections are performed primarily to detect electrical defects and anomalies present in counterfeit components. These tests are grouped into four categories. Parametric tests [43]–[45] are efficient at verifying IC's direct current (dc) and alternating current (ac) parameters. They can reveal the shift in electrical parameters due to components' prior usage or out of specification conditions. The functionality of a component can be checked by using functional tests [45], [46]. The defects and anomalies

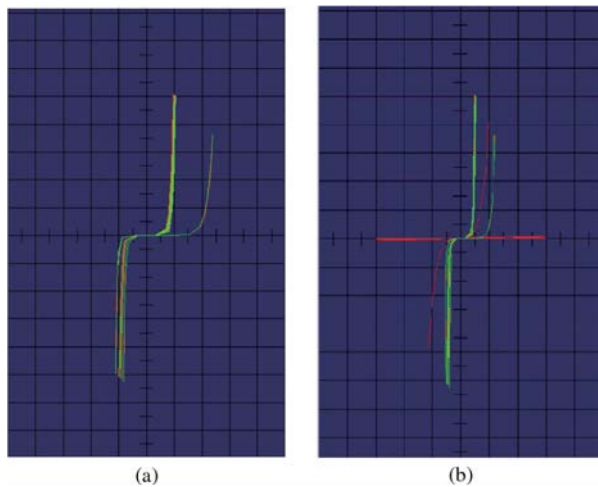


**Fig. 9. Counterfeit identification using curve trace [52].**

which impact the functionality of a component, such as broken/missing bond wires, cracks, damaged die, open/short interconnect etc., can be detected. In burn-in tests, the component is operated at stressed conditions, such as elevated temperature, to find infant mortality failures in order to assure reliability [47], [48]. In structural tests [49]–[51], test patterns are applied to a chip through internal scan chains to find the defects and anomalies related to the internal structures, interconnects, and logic gates. In this section, we will briefly describe some of the popular electrical inspections.

1) *Parametric Tests*: Parametric tests are performed over a range of operating temperatures to measure dc and ac parameters of a chip. They include curve tracing test, contact test, power consumption test, output short current test, output drive current test, threshold test, rise and fall time tests, setup, hold and release time tests, propagation delay tests, etc. The objective of parametric testing is to determine the quality of each product to avoid counterfeit distribution and production. This is accomplished by running a suite of tests or as many vital tests as possible to check the dc, ac, and parametric performance of the component in question. The intricacies of these tests can easily give test engineers a robust data set that they can use to uncover a counterfeit component where other test methodologies fail to uncover any problems or anomalies.

Fig. 9 illustrates an example of counterfeit identification using curve trace, with the curve of a genuine device shown in Fig. 9(a), and the curve of a counterfeit device shown in Fig. 9(b). It can be readily observed that the curve of counterfeit device differs significantly from that of the genuine device. Curve trace is also useful in verifying the device pinout and determining circuit damage that may have been created due to poor handling and electro-static discharge. The datasheet of the device can be used when

no genuine device is available for this test. We can further enhance the test capability by employing an automatic test equipment (ATE) to enable multiple parametric tests at one time, using automation to quickly perform measurements and evaluate the results.

Another form of proper parametric testing is to use an instrumentation board or instrumentation interface. These interfaces are used between the electronic component, standalone measurement equipment, and PC-based measurement equipment to provide specific parametric testing. These tests are either made available by the component manufacturer or are custom designed by the test lab with the end customer's review and approval.

2) *Functional Tests*: Functional tests are the most efficient way of verifying the functionality of a component and are perhaps one of the most expensive test methods available in the arsenal when testing complex devices. For instance, system memory chips will have to pass a series of functional tests exercising address, data lines, and bursting under various operating conditions (e.g., temperature, voltage, clock speed). A functional test could verify that all parts perform at specified higher frequencies and through the required temperature range using a functional baseline test. Testing over temperature or at room temperature (25 °C) are the industry standards for testing a circuit board design. This can be applied to the piece part(s) prior to manufacturing, but can be also used to test units on the manufacturing floor on assemblies to increase confidence that the assembled unit is free from faulty and counterfeit components, especially counterfeit components that were remarked as a higher performing part.

A counterfeit or substandard part may fail under a comprehensive functional test sequence. By checking that the device is functioning correctly, a whole range of other issues could be detected, e.g., wrong die, empty package, etc. The addition of a functional test process to current detection processes can be an effective way to increase counterfeit detection capability. A functional test will require a functioning system, most commonly a PCB-based system. For example, the system can be processor based with memory and a number of peripherals. The functional test is a software program with a series of algorithms that exercise and test specific elements of the design.

Several examples demonstrating the detection of counterfeit devices using functional tests under various operating conditions are shown in [53]. As an example, by lowering the ambient temperature in a temperature chamber while the device is repeatedly tested, failure was observed at very low temperature at  $-35^{\circ}\text{C}$ . However, the part was rated to operate at colder temperatures, indicating a defect in the device that could be counterfeit related.

### C. Aging-Based Statistical Fingerprints

During the lifetime of an IC, performances continuously degrade due to aging mechanisms. Using recycled

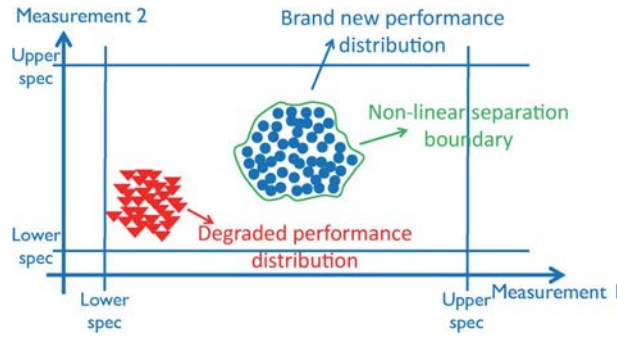


**Fig. 10. Nonlinear separation boundary for counterfeit device detection [16].**

counterfeit ICs as brand new will significantly reduce the capability of a device to perform its required functions for prolonged period of time. The most common aging phenomena include electromigration, negative bias temperature instability (NBTI), hot carrier injection (HCI), and time-dependent dielectric breakdown (TDDB). The following two methods leverage the impact of these aging phenomena in order to detect recycled counterfeit ICs.

1) *Early Failure Rate (EFR) Data Analysis*: In [16], a low-cost statistical approach was discussed to detect recycled counterfeit ICs by training a one-class classifier using only brand new devices. The measurements used to build the classifier are typical test results from production early failure rate (EFR) analysis required to release most products, such as  $V_{min}$ ,  $F_{max}$ , and  $I_{ddq}$ , thus no additional costs are incurred to perform identification. An overview of the one-class classifier, which is trained to separate new from aged devices in the space of such measurements, is illustrated in Fig. 10.

The first step involves collection of a set of parametric measurements, which can be taken from trustworthy provider across devices subject to process variations for the purpose of counterfeit IC detection. Then, using the parametric measurements of brand new devices, a one-class classifier is trained to distinguish counterfeit ICs from brand new ones. This approach is inspired by and resembles closely a machine-learning-based analog/RF IC test method [54].

The effectiveness of the approach is demonstrated on a microprocessor design, involving 49 parametric test measurements performed on 313 chips randomly chosen from different lots in production. These measurements are taken at five different time points for the same devices during burn-in test, in order to mimic the impact of aging degradation over time:  $t_0, t_1, t_2, t_3, t_4$ . Devices at  $t = t_0$  are referred to as brand new, while devices at  $t = t_i, i > 0$ , are referred to as counterfeit. Since the data set has a relatively high dimensionality  $d$  for this case study ( $d = 49$ ), a principal component analysis (PCA) [55] is performed in

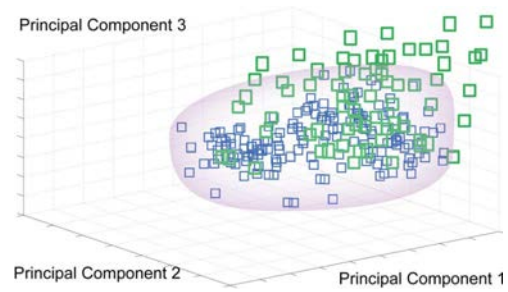


**Fig. 11. Projection of devices at  $t = t_0, t_1$ , shown by blue and green squares, respectively.**

order to map the original 49 measurements onto vectors in a lower dimensional space with cardinality  $d' < 49$ . The structure of the data is maintained while only nine principal components are kept, i.e.,  $d' = 9$ . Figs. 11 and 12 show the projection of devices at  $t = t_0, t_1$  and  $t = t_0, t_4$ , respectively, onto the first three principal components. As can be clearly observed, performance degradation caused by aging mechanisms is accelerated during the burn-in test. Indeed, an support vector machine (SVM) trained with half of the devices at time  $t = t_0$ , achieves 100% correct group classification rate for classifying devices at  $t = t_0, \dots, t_4$ , respectively, showing the excellent capability of detecting counterfeit devices using this approach [16].

2) *Circuit Path-Delay Analysis*: A path-delay fingerprinting technique, which was first introduced in [17] in the context of hardware Trojan detection, is adapted in [18] to distinguish recycled ICs from new ones. Due to degradation in the field, the path-delay distribution of recycled ICs will become different from that found in new ICs. Statistical data analysis can effectively separate the impact of process variations from aging effects on path delay. Simulation results demonstrate the efficiency of this technique for recycled IC identification.

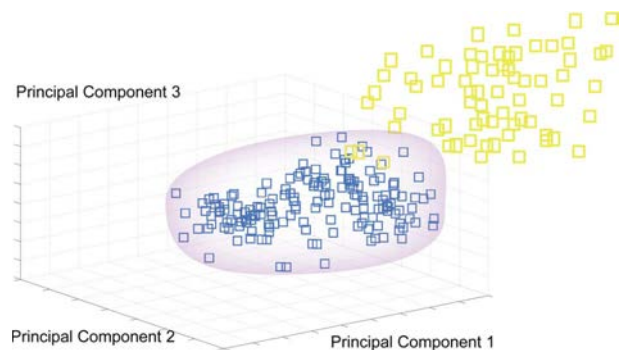


**Fig. 12. Projection of devices at  $t = t_0, t_4$ , shown by blue and yellow squares, respectively.**

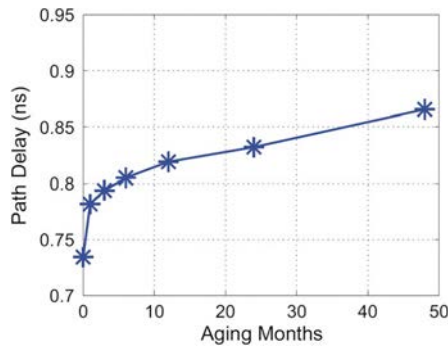


**Fig. 13.** Degradation of a critical path due to aging in a digital circuit [18].

Fig. 13 shows the delay degradation of a critical path from a random workload (functional patterns) applied to the primary input of a digital circuit. The path was aged for four years, using simulation, with NBTI and HCI effects at room temperature. We can observe from Fig. 13 that the degradation of the path used for one year is around 10% while if the circuit is used for four years, the degradation is about 17%, indicating that most aging occurred at the early usage phase of the circuit.

The approach in [18] consists of three major steps. First, paths are simulated and selected according to their aging rate. Next, the delay of these paths is measured by a clock sweeping technique in new ICs (either during manufacturing test on all ICs or during authentication on a sample of new ICs) and in any available devices under authentication.

Statistical analysis is used to decide whether the device under authentication is a recycled IC. The same test patterns will be applied to the circuit under authentication, taken from the market, in a near-identical environment. The path-delay information of the circuit will then be processed by the same statistical data analysis methods. In a simple analysis, if the fingerprint of the circuit is outside the range of the new ICs' fingerprint, there is a

high probability that the circuit is a recycled IC. Otherwise, it is likely a new IC. The longer the circuit has been used in the field, the more aging effects it will have experienced, making it easier to identify.

The recycled IC identification flow was demonstrated using 45-nm technology on a few benchmarks. HSPICE MOSRA [56] is used to simulate the effects of aging on the path delay of different benchmarks. Fig. 14 shows the PCA results of the ICs used for three months with process variation. The used ICs are represented by red dots; the convex hull is built up from new ICs' data, and represents the fingerprint for new ICs. It can be observed that the used ICs are completely separated from the signature of the new ICs, implying a 100% detection rate for recycled ICs.

We note that in cases where the counterfeit ICs are affected by large process variation, detecting these counterfeit ICs using statistical methods may be challenging, since patterns of counterfeit chips and golden chips tend to overlap in the space of parametric and side-channel measurements. To avoid/mitigate the impacts of large process variations, techniques proposed to detect hardware Trojans can be adopted for counterfeit detection purpose. In [57], advanced signal processing techniques based on Karhunen-Loève expansion are used to find a signal subspace from which the process noise is absent in order to identify Trojans that are well hidden within the variations of the signals generated by the process noise. In [58], it is shown that using postsilicon multimodal thermal and power characterization techniques, one can significantly improve the Trojan detection sensitivity, even under large process variations. In [59], it is demonstrated that measuring currents locally and from multiple power ports or pads can greatly enhance Trojan detection. A region-based transient-power signal analysis method to reduce the impact of increasing process variation in detecting hardware Trojans was discussed in [60]. It is shown that using signal calibration techniques, one can further increase the distance between Trojan-free and Trojan-infested circuits under different process parameters.

#### D. Counterfeit Detection Summary

The production and distribution of counterfeit parts are rising, and more of such parts are finding their way into consumer and military devices. As counterfeiters become more sophisticated, so must the methods used for detecting counterfeit parts. Over the last decade, multiple methods have been successfully utilized to detect counterfeit parts in an attempt to keep them from being used in final assemblies.

Table 4 summarizes different types of counterfeits and detection methods. As seen, there is no single test which can work to detect all counterfeit components. The parts themselves are different, the uses are varied, and, therefore the test plan for each component must be customized. A full suite of tests is required to identify counterfeit components. The functional and parametric tests have been vital in testing electronic components for over 40 years and are a

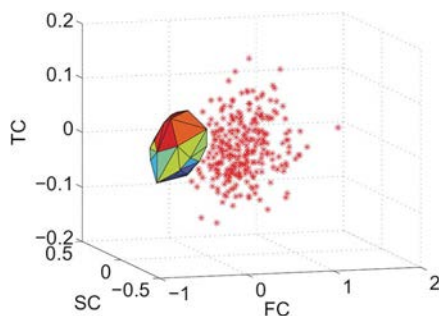


**Fig. 14.** PCA results of the ICs used for three months with process variation [18].

Table 4 Summary of Counterfeit Detection Methods

| Detection Methods                  | Recycled | Remarked | Overproduced | Out-of-spec/Defective | Cloned |
|------------------------------------|----------|----------|--------------|-----------------------|--------|
| Low Power Visual Inspection (LPVI) | Low      | Low      | NA           | NA                    | NA     |
| X-Ray                              | Low      | NA       | NA           | NA                    | NA     |
| Microblast Analysis                | Low      | Medium   | NA           | NA                    | NA     |
| Scanning Acoustic Microscopy (SAM) | Medium   | NA       | NA           | NA                    | NA     |
| Scanning Electron Microscopy (SEM) | Medium   | Medium   | NA           | NA                    | Low    |
| Material Analysis                  | Medium   | Medium   | NA           | NA                    | NA     |
| Parametric Tests                   | Medium   | Low      | Low          | Medium                | Low    |
| Functional Tests                   | Medium   | Low      | Low          | Medium                | Low    |
| EFR Analysis                       | Medium   | Medium   | Low          | Low                   | Low    |
| Path-Delay Analysis                | Medium   | Medium   | Low          | Low                   | Low    |

\* Forged documentation and tampered types are not discussed here because of their unique detection challenges.

paramount tool for containing the counterfeit component epidemic plaguing our industry. However, in the electronics industry, it may not be practical cost-wise and time-wise to test every single functional and parametric requirement of the manufacturer. Proper strategic planning and outlining of tests between test lab and clients are important to establish an objective of performance measurement. Highly sophisticated counterfeit components may work under a subset of operating conditions but fail under others. Testing over a range of different test methods on the device will provide the highest level of assurance that electronic components are genuine. It is important to note that additional research is needed in early failure rate (EFR) data analysis and circuit path-delay analysis to verify the effectiveness of these methods for determining if a used part is sold as new. Preliminary analysis and results from these methods are encouraging.

### III. COUNTERFEIT AVOIDANCE

Detection of counterfeit components is a major challenge because of the excessive test time, cost, and lack of metrics to evaluate the test confidence in evolving area of concern due to the rapidly changing threat environment. The issue urgently necessitates the development of innovative

avoidance mechanisms to be incorporated in the design. These measures help detecting suspect parts without the need for aforementioned expensive detection methods. In this section, we will briefly describe some currently available avoidance methods.

#### A. CDIR Sensors

The combating die and IC recycling (CDIR) sensor was developed to prevent parts from recycling [26], [27]. The authors proposed three different structures to implement CDIR sensors. The first technique inserts a lightweight ring oscillator (RO)-based sensor in the chip to capture the usage of the chip in the field and provides an easy detection capability. Fig. 15 shows the structure for this RO-based sensor. This sensor is composed of a reference ring oscillator (reference RO) and a stressed ring oscillator (stressed RO). The sensor relies on the aging effects of MOSFETs to change the RO frequencies. The difference between the frequency of reference and stressed ROs gives the approximate usage time of the chip in the field.

The other two structures are antifuse-based sensors which are composed of counters and an embedded antifuse memory block. The counters are used to record the usage time of ICs while their values are continuously stored in an

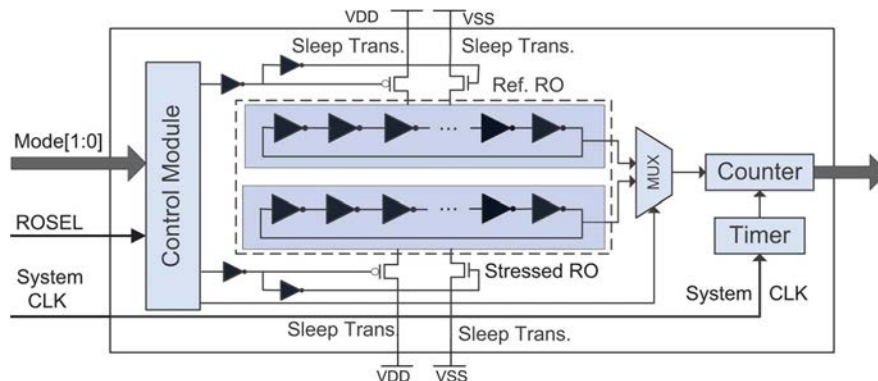


Fig. 15. RO-based CDIR sensor [26].



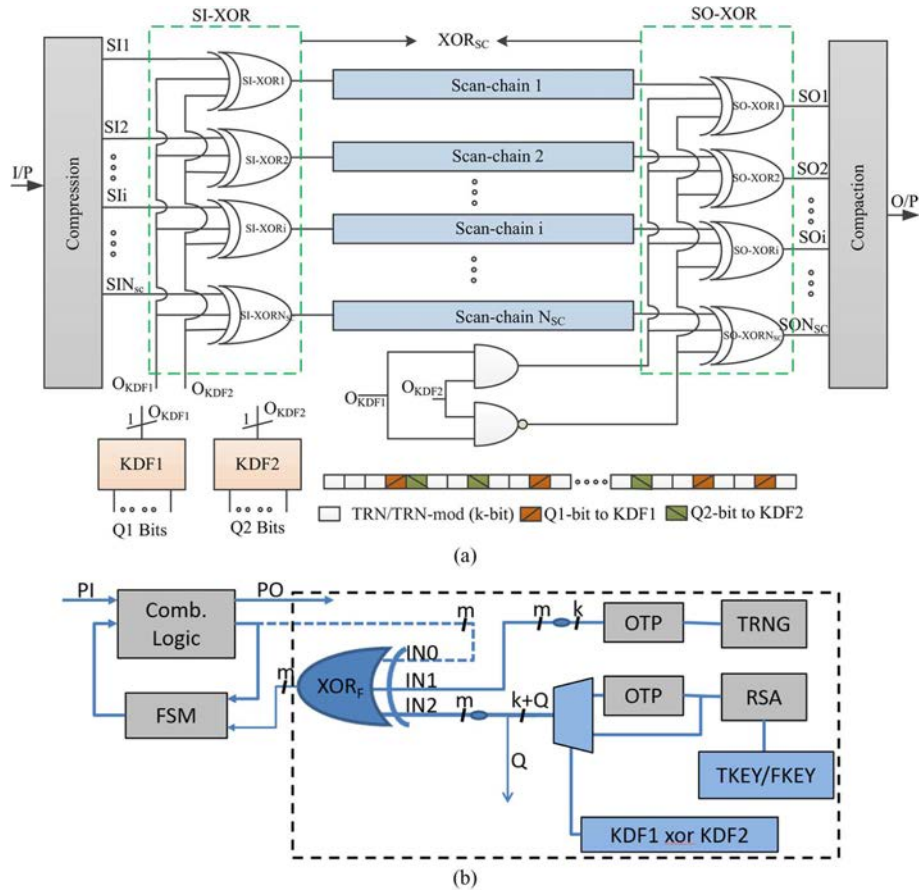


**Fig. 18. Internal architecture of SST inside ICs [19]. (a) Scan locking block. (b) Functional-locking block with  $m = pk$  expansion.**

some untrusted foundries and assemblies can sell defective, out-of-spec, or even overproduced ICs on the open market. Secure split test (SST) secures the manufacturing test process to prevent counterfeits, allowing the design house to protect and meter their IPs [19]. SST introduces hardware components for cryptography and locking mechanisms to block the correct functionality of an IC until it is activated by the IP owner during or after the test. Thus, SST brings design houses back into the manufacturing test process.

Fig. 17 shows the communication flow between IP owner, foundry, and assembly. The IP owner first obtains a die's true random number (TRN) from foundry and modifies it in such a way that only he knows it. A test key (TKEY) is then generated from this modified TRN and the IP owner's private key. The IP owner sends the TKEY to the foundry for each die. The chip then encrypts the modified TRN by using a public key and tests the die using this modified TRN. It then sends the response back to the IP owner for the pass/fail decision. After verifying the die, the foundry sends it to the assembly. After packaging the die, the assembly tests the parts again. The assembly then sends the response to the IP owner. The IP owner then

unlocks the good ICs using the final key (FKEY) and sends them to the market.

Fig. 18 shows the SST architecture. It secures the test processes of ICs by inserting scan locking block and functional-locking block in the design. The details of a scan-locking block are presented in Fig. 18(a). In scan-locking block, the inputs to some scan chains are inverted when FKEY is applied; the input bits are transparent when TKEY is applied. At the same time, the output from some of the scan chains can be inverted when TKEY is applied and transparent otherwise. This block is introduced so an attacker cannot extract any information from an unlocked IC. The scan-locking block consists of three-input xor gates inserted at scan chain inputs (SI-xor) and outputs (SO-xor) and two key-determining functions (KDFs). Each KDF is composed of xor gates forming an xor odd-function circuit which outputs 1 if the input is odd (odd number of 1s) or 0 if it is even (even or zero number of 1s). Two KDFs, KDF1 and KDF2, are used to detect the type of key that has been provided by IP owner and determine the function of the scan-locking mechanism. The outputs of KDFs, OKDF1 and OKDF2, are fed to the three-input xor gates through some logic gates to make the xors

transparent (when  $KDF1 = KDF2$ ) or inverting (when  $KDF1 \neq KDF2$ ). The  $Q$  bits needed to control each KDF are added to TRN and TRN-mod by the IP owner.  $Q$  is divided into two parts  $Q1$  and  $Q2$  as inputs for KDF1 and KDF2, respectively. The position of  $Q$  additional bits is random and known only by the IP owner since TRN-mod and TRN are encrypted and only TKEY and FKEY are visible by the foundry.

The functional-locking block's purpose is to ensure that only unlocked ICs will have the correct functionality. ICs will only function correctly when the IP owner sends the correct functional key. The functional-locking block is made up of three smaller hardware blocks, an  $xor_F$  mask for functional locking, true random number generator (TRNG), and RSA decryption logic, as shown in Fig. 18(b). SST inserts the  $xor_F$  mask on noncritical paths in the circuit. These xors have three inputs connected to circuit paths (IN0), TRNG output (IN1), and RSA output (IN2). An  $xor_F$  is transparent only when both inputs coming from the TRNG and RSA outputs are the same; otherwise it acts as an inverter. The TRNG output is different for each IC but it is constant throughout the IC's lifetime since it is stored in a onetime programmable (OTP) memory. The RSA component receives an encrypted key from the IP owner; this key is decrypted and connected to IN2. The circuit only becomes functional when the appropriate RSA input is applied; this gives the IP owner control over when/whether to activate the IC. The encryption and transmission of the key make up the communication component of SST. A server owned by the IP owner receives TRNs for each die and creates test keys (TKEYs); it then receives test outputs and compares them to the expected outputs to determine which dies have passed or failed. The same procedure is followed during assembly. The server decides whether an IC has passed the necessary tests after assembly, and only then, it sends the functional key (FKEY) to activate and unlock the IC.

Different types of attacks have been analyzed in [19]. SST is very efficient in preventing overproduced ICs as the design house unlocks the number of ICs they want. As the decision of the test process is taken by the design house, the foundry cannot source defective/out-of-spec ICs in the supply chain. Cloned ICs can also be detected by checking the registered ID of the IC in design house's secured database.

### C. Hardware Metering

Hardware metering places a set of security protocols that enable the design house to achieve postfabrication control of the produced ICs. This method provides the design house with a unique way to identify each IC produced with the same masks [21], [62]. Hardware metering is broadly categorized into passive or active types. Early passive metering includes indented and digitally stored serial numbers on the ICs as a nonfunctional identification method. In functional identification, PUFs are introduced to uniquely identify each IC and register the IC using

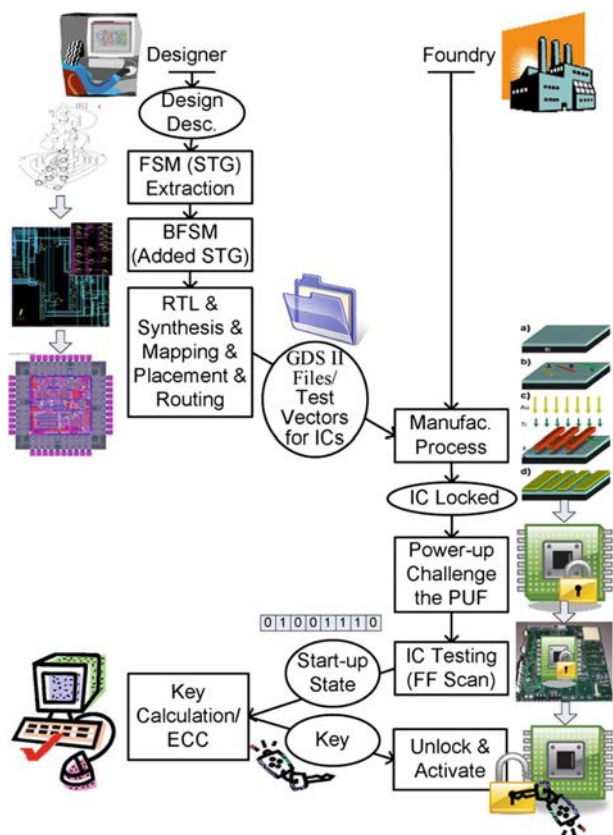


Fig. 19. IC enabling flow by active metering [68].

challenge–response pairs to prevent cloning and removal attacks. The ICs are authenticated by searching the response stored in the challenge–response pair database [62]–[67].

In addition to unique identification, active metering approaches lock each IC until it is unlocked by the IP holder [20], [69]–[73]. This locking is mostly done in three ways: 1) initializing ICs to a locked state on power-up [20]; 2) combinational locking by scattering  $xor$  gates randomly throughout the design [71]–[73]; and 3) adding a finite-state machine (FSM) which is initially locked and can be unlocked only with the correct sequence of primary inputs [70], [74].

In [75] and [76], a logic encryption technique is used to hardwire designs with built-in keys that are unique to each IC, and ensures that the application of any invalid key on the protected design forces the design to produce incorrect results. Logic encryption inserts additional circuit elements into the original design. It has been shown in [75] that testing concepts such as fault activation, propagation, and masking can be utilized to guide the insertion of key gates. This way, a perfect control over the functional corruption due to invalid key application can be achieved. The same concepts can not only be utilized by an attacker to leak the secret logic encryption key, but also by a

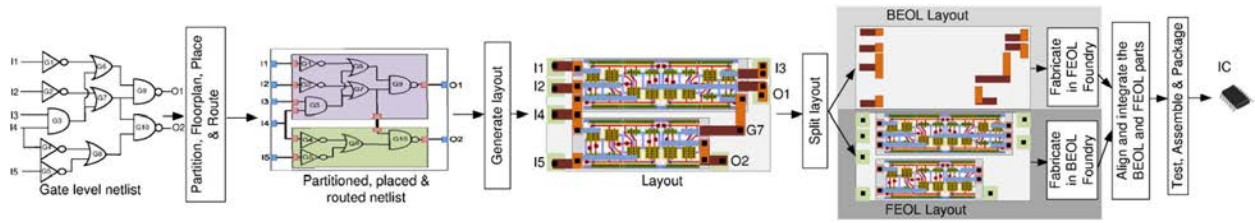


**Fig. 20. Split-manufacturing-aware design flow [80].**

designer to guide the insertion of key gates in attaining a hard-to-break logic encryption [76]. Fig. 19 shows an IC enabling flow by active metering [77].

In [78], Barak *et al.* lay the theoretical groundwork for adopting obfuscation as a means for protecting programs. An obfuscator  $\mathcal{O}$  is defined as a “compiler” that takes as input a program (or circuit)  $P$  and produces a new program  $\mathcal{O}(P)$  that has the same functionality as  $P$  yet is “unintelligible” in some sense. Obfuscators would have a wide variety of cryptographic and complexity-theoretic applications, ranging from software protection to homomorphic encryption. Most of these applications are based on an interpretation of the “unintelligibility” condition in obfuscation as meaning that  $\mathcal{O}(P)$  is a “virtual black box,” in the sense that anything one can efficiently compute given  $\mathcal{O}(P)$ , one could also efficiently compute given oracle access to  $P$ .

#### D. Split Manufacturing

Globalization of IC design flow has led to several security vulnerabilities. In order to mitigate the risks in manufacturing, split manufacturing approach has been proposed [79]. In this approach, the layout of the design is split into the front end of line (FEOL) layers and back end of line (BEOL) layers which are then fabricated separately in different foundries. The FEOL layers consist of transistors and other lower metal layers and the BEOL layers consist of the top metal layers. Postfabrication, the FEOL and BEOL wafers are aligned and integrated together using

either electrical, mechanical, or optical alignment techniques. The final ICs are tested upon integration of the FEOL and BEOL wafers. The asymmetrical nature of the metal layers facilitates split manufacturing [80].

Fig. 20 shows an example of split-manufacturing-aware IC design flow, as depicted in [80]. A gate level netlist is partitioned into blocks which are then floorplanned and placed. The transistors and wires inside a block form the FEOL layers. The top metal wires connecting the blocks and the IO ports form the BEOL layers. The layout of the entire design is then split into FEOL and BEOL layers. The two layouts are then fabricated in two different foundries. Finally, FEOL and BEOL layouts are integrated by using electrical, mechanical, or optical alignment techniques and tested for defects.

#### E. IC Camouflaging

During the manufacturing, the chips can also be cloned in an unauthorized production by reverse engineering the original design or pirating the IP. In an effort to hamper an attacker from reverse engineering a chip, IC camouflaging has been proposed by introducing dummy contacts into the layout [81]. By using a mix of real and dummy contacts, one can camouflage a standard cell whose functionality can be one of many. If an attacker cannot resolve the functionality of a camouflaged gate, he/she will extract an incorrect netlist.

In one embodiment of IC camouflaging, the layouts of logic gates are designed to look identical, resulting in an incorrect extraction. To thwart reverse engineering of an

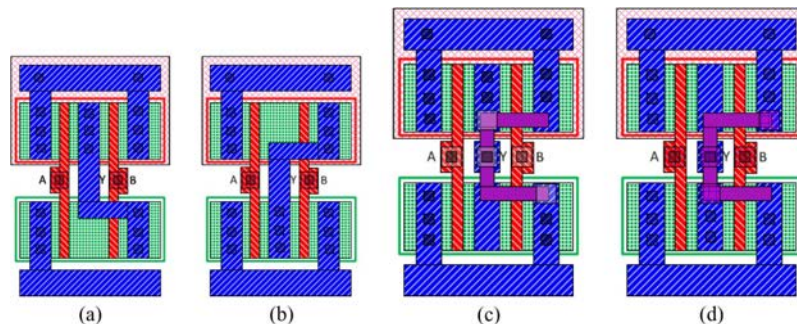


**Fig. 21. Example of IC camouflaging: standard cell layout of regular two-input (a) NAND and (b) NOR gates and camouflaged NAND (c) and NOR (d) gates [82].**

IC, any camouflaging technique has to provide resiliency to reverse engineering and corrupted outputs. The reverse engineer's inability to partially etch a layer is the basis for dummy contacts-based camouflaging. Contacts are conducting materials that connect two adjacent metal layers or a metal layer 1 and poly. They pass through the dielectric that separates the two connecting layers. While the true contact has no gap, a dummy contact has a gap in the middle and fakes a connection between the layers [82].

Fig. 21 shows an example of IC camouflaging as depicted in [82]. The layout of regular NAND and NOR cell shown in Fig. 21(a) and (b) looks different and is hence easy to reverse engineer. However, the layout of camouflaged NAND and NOR cell shown in Fig. 21(c) and (d) looks identical and is difficult to differentiate. When deceived into incorrectly interpreting the functionality of the camouflaged gate, the attacker may obtain a reverse-engineered netlist that is different from the original. The netlist obtained by an attacker is the deceiving netlist where the functionality of the camouflaged gates is arbitrarily assigned.

#### F. Hardware Watermarking

Hardware watermarking [83] has received much attention in the recent years to secure hardware intellectual properties (IPs) used in high-density ICs (e.g., system on chips). The reuse of these IPs poses a great concern to the industry as the infringement of IPs, e.g., trademark, copyright, or patent violation during the design. Hardware watermarking uniquely identifies an IP by creating a unique fingerprint in it. In recent years, different watermarking techniques have been proposed. Constraint watermarking affects the IP core at the GDSII level. The goal of constraint watermarking is to create a physical design pattern that can be demonstrated to but not be replicated purely by coincidence [84]. This is referred to as “proof of authorship,” the probability that the occurrence of a watermark on a known nonwatermarked IP core is purely coincidental. Watermarks can be inserted into the bitstream of an IP core by injecting watermark bits into unused combinational logic block outputs. Another implementation of constraint watermarking involves breaking paths of logic into subpaths, each with unique timing constraints that add up to the timing constraint of the original path. In [85], Castillo *et al.* propose a hardware description language (HDL) level watermarking. They store the digital signature within memory structures or combinational logic that are part of the system, at the high level description of the design. In [86], Lach *et al.* propose a cryptographically encoded signatures to be placed in a field-programmable gate array (FPGA) design component to identify original recipient. Another watermarking method for IP protection was proposed at the combinational-logic-synthesis level [87].

#### G. Physical Unclonable Functions

Reverse-engineering tools such as microprobing, laser cutting, glitch attacks, and power analysis have become very

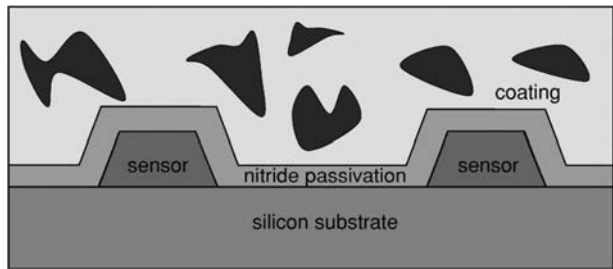


**Fig. 22. Structure of a coating PUF [88].**

advanced to date and allow an attacker to read and copy the stored encrypted information in a device. To overcome this challenge, hardware intrinsic security (HIS) has been proposed as a mechanism that can provide security based on inherent properties of an electronic device [23]. PUFs [24], for example, belong to the category of such HIS mechanisms. PUFs aim to measure the responses of hardware to certain given inputs, which depend on the unique physical properties of the device, since process variations affect each device in a unique and unclonable fashion.

Rather than creating a single encrypted key, PUFs implement challenge–response authentication. When a stimulus is applied to the device, it reacts in an unpredictable (but repeatable) way due to the complex interaction of the stimulus with the physical structure of the device. This exact structure depends on physical factors introduced during manufacture which are unpredictable. The applied stimulus is called the challenge, and the reaction of the PUF is called the response [24]. A specific challenge and its corresponding response together form a challenge–response pair.

Various methods have been proposed to obtain this challenge–response pair. In [24], optical characteristics are exploited by using the speckle patterns of optical medium of laser light. The PUF consists of a transparent material that is doped with light scattering particles. When a laser beam shines on the material, a random and unique speckle pattern will arise. The placement of the light scattering particles is an uncontrolled process, and the interaction between the laser and the particles is very complex. Therefore, it is very hard to duplicate the optical PUF such that the same speckle pattern will arise.

A coating PUF was proposed in [88], where a network of metal wires is laid out in a comb shape in the top layer of an IC. The space between and above the comb structure are filled with an opaque material and randomly doped with dielectric particles. Fig. 22 illustrates the structure of a coating PUF, as proposed in [88]. The manufacture of the coating is an unpredictable mixing process, therefore, size and dielectric strength of the particles will be random up to a certain extent. Consequently, the measured capacitance values are unpredictable. This unique randomness can be used to obtain a unique identifier for the device carrying the coating PUF. Moreover, the placement of this

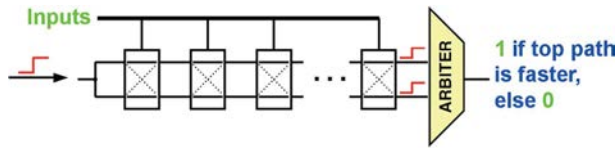


**Fig. 23. Delay PUF circuit: two delay paths with the same layout length for each input, and an output based on which path is faster [63], [91], [92].**

opaque PUF in the top layer of an IC protects the underlying circuits from being inspected by an attacker, e.g., for reverse engineering. When an attacker tries to remove (a part of) the coating, the capacitance between the wires is bound to change and the original unique identifier will be destroyed. In [89], an unclonable RFID tag is built using coating PUFs.

In [63], [90], and [91], a delay PUF exploiting the random variations in delays of wires and gates on silicon was proposed. Given an input challenge, a race condition is set up in the circuit, and two transitions that propagate along different paths are compared to see which comes first. An arbiter, typically implemented as a latch, produces a 1 or a 0, depending on which transition comes first. Fig. 23 illustrates the implementation of delay PUFs. The inputs determine the delay path of each switching block shown by rectangles in Fig. 23. The switching block is typically made by a pair of multiplexers controlled by the same input bit. The signal goes through only one path at a time in the block, which is controlled by the input bit. In this way, the circuit can create a pair of delay paths for each input combination.

The authentication of delay PUFs is illustrated in Fig. 24. As explained in [63], the PUF can have an exponential number of challenge–response pairs where the response is unique for each IC and each challenge. These unpredicted responses can be stored by a trusted party in a database for future authentication operations. To check the authenticity of an IC later, the trusted party selects a challenge that has been previously recorded but has never been used for an authentication operation, and obtains the PUF response from the IC. If the response matches the previous recorded one, the IC is then considered authentic.

A PUF based on ring oscillators was also discussed in [63]. A secure processor design using a PUF was shown in

[93]. In [92], it was shown that the delay PUFs can be integrated into a small passive RFID tag for anticounterfeiting and secure access.

Other types of PUF implementation include SRAM PUFs [94], butterfly PUFs [64], and bistable ring PUFs [95], [96].

## H. Package ID

The counterfeit avoidance measures discussed so far can only be implemented on dies that only target new large ICs. However, there is a wide variety of components that need to be addressed for counterfeit avoidance. These components mainly belong to: 1) large and small new analog; 2) small new digital; and 3) all active and obsolete categories. There are several challenges for the implementation of chip IDs in these designs. First, there is not enough space for adding any extra hardware to the designs. Second, one does not have the authority and option to make changes in the masks for active components. Finally, obsolete components are no longer manufactured. For tagging such active and obsolete components, we need to create package IDs that do not require access to designs. No package modifications are allowed during the generation of package IDs. DNA markings and nanorods are the viable options to create package IDs.

In DNA markings, a unique genetic sequence is generated by scrambling the plant DNA and mixing these new sequences with inks. These inks are then applied on the packages of the ICs at the end of the packaging process for new components. The active and obsolete components are authenticated first and then this DNA marking is placed on the package. Authentication includes first checking whether the ink fluoresces under specific light, and second sending a sample of the ink to a lab to verify that the DNA is in the database of valid sequences [28]. Recently, Department of Defense (DOD) mandated [97] that the DNA marking be placed on the components in order to track them throughout the supply chain. In nanorods, a microscopic pattern is created by growing an array of nanospheres to make nanorods that are less than 100 nm long [29]. Each time the process is repeated, the same pattern is created, but the exact angle and length of each individual nanorod varies, so that each set of nanorods is distinct. After the array of nanorods is grown, it is applied to a chip using a specialized printer. The chip can be authenticated by comparing the overall pattern and visual properties of each nanorod to a database.

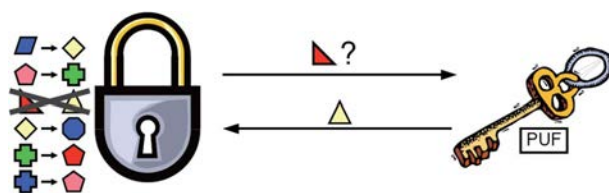


**Fig. 24. Delay PUF authentication [63], [91], [92].**

## I. Summary of Counterfeit Avoidance Methods

Table 5 shows the summary of counterfeit avoidance methods. The first column represents avoidance methods discussed in this paper. Columns 2–6 and 7–9 show counterfeit types and component types, respectively. Each entry in columns 2–6 represents how effective the methods are at detecting that counterfeit type. Each entry in columns 7–9 shows target component types with avoidance effectiveness. CDIR sensors can detect only recycled and remarked

Table 5 Summary of Counterfeit Avoidance Methods

| Avoidance Methods                      | Counterfeit Types* |          |              |                           |        | Component Types |        |        |
|----------------------------------------|--------------------|----------|--------------|---------------------------|--------|-----------------|--------|--------|
|                                        | Recycled           | Remarked | Overproduced | Out-of-spec/<br>Defective | Cloned | Obsolete        | Active | New    |
| CDIR Sensors                           | High               | High     | NA           | NA                        | NA     | NA              | NA     | High   |
| Secure Split Test (SST)                | NA                 | NA       | High         | High                      | Medium | NA              | NA     | Medium |
| Hardware metering                      | NA                 | NA       | Low          | NA                        | Low    | NA              | NA     | Medium |
| Split Manufacturing                    | NA                 | NA       | Low          | NA                        | Low    | NA              | NA     | Medium |
| IC Camouflaging                        | NA                 | NA       | NA           | NA                        | Low    | NA              | NA     | Medium |
| Hardware Watermarking                  | NA                 | NA       | NA           | NA                        | Medium | NA              | NA     | Medium |
| Physically Unclonable Functions (PUFs) | NA                 | NA       | Low          | NA                        | Low    | NA              | NA     | Medium |
| Package ID                             | Medium             | Medium   | NA           | NA                        | NA     | High            | High   | High   |

\* Forged documentation and tampered types are not discussed here because of their unique avoidance challenges.

new components. SSTs can target overproduced, cloned, and out-of-spec/defective parts. Hardware metering and PUFs can detect only cloned and overproduced components, while package ID can aim only for recycled and remarked types. IC camouflaging and hardware watermarking can be used to detect only cloned ICs. Overproduced and cloned ICs can be detected by split manufacturing. All these avoidance methods, except package ID, can target only new components. While many of the counterfeit avoidance methods could be useful for resolving the counterfeit problem, many of the techniques are not deployed in a production environment. Many of the techniques are promising, but need further research and development.

#### IV. DETECTION AND AVOIDANCE CHALLENGES

The limited research and development in the domain of counterfeit detection and avoidance has left major challenges to be addressed. In addition, the need to addressing this problem requires immediate action by industry, government, and academia since the impact of counterfeit electronics will not only be seen on business but also on our daily life. Lack of innovative and comprehensive solutions to detect counterfeit parts could be catastrophic.

The challenges we deal with today arise from the fact that: 1) the number of counterfeit parts entering the electronic component supply chain is increasing; 2) the sophistication of the techniques used by the counterfeiters is increasing; 3) there are different types of electronics components and one solution will not address this challenging problem for all types of counterfeit parts and types; 4) multiple test methods are required to detect the multiple known anomalies and are proven ineffective for detecting all types of counterfeit parts; 5) there is a lack of sufficient support to academia for developing innovative detection and avoidance solutions; and 6) there is a lack of sufficient amount of counterfeit data (types, defects, test cost, test time, etc.).

As a consequence, today, we cannot identify a part as counterfeit with a very high level of confidence without

performing multiple test methods and/or receiving input from the original component manufacturer to assist in providing needed data, verifying the results and assisting in the final conclusion. There is clearly a need for the development of anticounterfeit mechanisms during design of the ICs. New methods are needed to allow track and trace of the components in the supply chain. Metrics are needed to evaluate the effectiveness of the existing test methods. New data collection mechanisms are needed to continuously monitor counterfeit activities and help the community understand the current trends and new threats. The processes defined must ensure detection, but must also balance the cost and lead-time issues associated with destruction of components, and nonrecurring engineering costs and testing. Sophisticated counterfeits may not be detected by a simple external, physical inspection process. New test techniques are needed to detect all defects and anomalies associated with counterfeit parts [15] in a comprehensive fashion.

##### A. Detection Challenges

Physical inspections suffer from several challenges. First, many of them are destructive; sample preparation is extremely important as it directly relates to the test confidence. The chance of selecting a counterfeit component from a lot is extremely small. Second, test time and cost are major limiting factors in the use of physical tests for counterfeit detection. For some tests, it may take up to several hours (e.g., typically more than eight hours for SEM analysis or to perform functional electrical tests that includes set-up) to test a single component. Third, most of the tests are carried out without automation and with no metrics for quantifying against a set of counterfeit types, anomalies, and defects. The test results are mostly dependent on the subject matter experts (SMEs) interpretation of results to distinguish between quality issues and counterfeit issues. The decision-making process is then error prone as it entirely depends on the operator or SMEs. It has been demonstrated by the G-19A group that a chip was marked as counterfeit in one lab while it was called authentic in another lab [98].

There are several limitations that make electrical tests ineffective. Increased process variations and environmental variations in lower technology nodes make the parametric test results indeterminate as the electrical parameters of a component vary significantly. Obsolete and active parts make the functional tests ineffective as test program generation for those parts with limited knowledge of functionality is extremely difficult if not impossible to obtain data or manufacturer test fixtures. The design and test information for the obsolete parts may be lost, or the OCM may no longer exist, as the design may have been sunset decades ago. Burn-in tests are helpful in detecting infant mortality failures of components, but are extremely expensive and time consuming. These tests are attractive and useful only for critical and high-risk, high-reliability applications where obtaining a high test coverage is of prime importance because of excessive test time and cost. Total access to the internal scan chains of a component is required for structural tests to be effective. Generally, IP owners disable the scan chains by using fuses to prevent the outside world from having access to their designs. Even if the scan is accessible, limited knowledge of the chip design and layout makes such test ineffective. Moreover, for obsolete parts, design for testability (DFT) may not even exist.

## B. Avoidance Challenges

Counterfeit avoidance techniques, described in Section III, are still a work-in-progress and they pose unique challenges that must be addressed before deployment into the high-reliability systems. For instance, in the hardware metering technique, the untrusted foundry can fabricate more ICs while pretending the yield is low. That allows them to put more functioning chips into the market. A design house cannot prevent out-of-spec and defective ICs from entering the supply chain through an untrusted access point. As for PUFs, reliability is the major concern that must be addressed. PUFs are proven to be sensitive to a wide range of environmental variations (temperature, power supply noise), and aging.

DNA markings suffer from several limitations that introduce serious concerns about their applicability in coun-

terfeit avoidance at the large scale. The fast authentication achieved by observing the fluorescence of markings under specific light can potentially be imitated by counterfeiters, either by invalid DNA or by other materials. The detailed DNA validation process is extremely time consuming and costly [99]. For fast authentication, nanorods must also deal with the same issues as in DNA markings. The reliability of the materials used by both methods cannot be completely verified.

The avoidance solutions must also consider the part types (analog, digital), the part size, the applications risks, cost of inserting the anticounterfeit measures, potential attacks, etc.

## V. CONCLUSION

Detection and/or prevention of counterfeit electronic components have become a major challenge in the electronic component supply chain. In this paper, we first presented the various counterfeit types currently present in the supply chain, followed by a taxonomy of counterfeit detection methods which describes existing capabilities for counterfeit detection. We also briefly described some well-known physical and electrical inspection methods. We then presented counterfeit avoidance measures to emphasize what needs to be done in order to detect these counterfeit components in a proactive, rather than reactive manner, if such measures were to be in place. We presented the key challenges in counterfeit detection and avoidance, as well as contemporary research opportunities. ■

## Acknowledgment

The authors would like to thank Steve Walters of Honeywell (Clearwater, FL, USA) and Sultan Lilani of Integra Technologies (Wichita, KS, USA) for providing their valuable feedback on physical inspection methods. They would also like to thank Prof. M. Anwar of the University of Connecticut (Storrs, CT, USA) for his comments on physical inspections.

## REFERENCES

- [1] U.S. Senate Committee on Armed Services, "Inquiry into counterfeit electronic parts in the Department of Defence supply chain," May 2012. [Online]. Available: <http://www.armed-services.senate.gov/Publications/Counterfeit%20Electronic%20Parts.pdf>
- [2] U.S. Department of Commerce, "Defense industrial base assessment: Counterfeit electronics," Jan. 2010.
- [3] M. Pecht and S. Tiku, "Bogus: Electronic manufacturing and consumers confront a rising tide of counterfeit electronics," *IEEE Spectrum*, vol. 43, no. 5, pp. 37–46, May 2006.
- [4] N. Kae-Nune and S. Pessegueir, "Qualification and testing process to implement anti-counterfeiting technologies into IC packages," in *Proc. Design Autom. Test Eur. Conf.*, 2013, pp. 1131–1136.
- [5] S. Bastia, "Next generation technologies to combat counterfeiting of electronic components," *IEEE Trans. Compon. Packag. Technol.*, vol. 25, no. 1, pp. 175–176, Mar. 2006.
- [6] IHS, "Reports of counterfeit parts quadruple since 2009, challenging U.S. Defence Industry and National Security," Apr. 2012. [Online]. Available: <http://www.ihs.com/images/IHS-iSuppli-Reports-Counterfeit-Parts-Quadruple-Since-2009.pdf>
- [7] U. Guin, M. Tehranipoor, D. DiMase, and M. Megrđician, "Counterfeit IC detection and challenges ahead," in *ACM SIGDA E-Newslett.*, vol. 43, no. 3, Mar. 2013.
- [8] Semiconductor Industry Association (SIA), "Winning the battle against counterfeit semiconductor products," Aug. 2013.
- [9] S. Nevison, "Counterfeit parts infiltrate aerospace projects," *Industry Market Trends*, Apr. 2009.
- [10] IHS Technology, "Top 5 most counterfeited parts represent a \$169 billion potential challenge for global semiconductor market," Apr. 4, 2012. [Online]. Available: [http://www.isuppli.com/Semiconductor-Value-Chain/News/pages/Top-5-Most-Counterfeited-Parts-Represent-a-\\$169-Billion-Potential-Challenge-for-Global-Semiconductor-Market.aspx](http://www.isuppli.com/Semiconductor-Value-Chain/News/pages/Top-5-Most-Counterfeited-Parts-Represent-a-$169-Billion-Potential-Challenge-for-Global-Semiconductor-Market.aspx)
- [11] D. Pantic, "Benefits of integrated-circuit burn-in to obtain high reliability parts,"

- IEEE Trans. Reliab.*, vol. 35, no. 1, pp. 3–6, Apr. 1986.
- [12] J. Carulli and T. Anderson, “Test connections—Tying application to process,” in *Proc. IEEE Int. Test Conf.*, 2005, DOI: 10.1109/TEST.2005.1584030.
  - [13] Aerospace Industries Association, “Counterfeit parts: Increasing awareness and developing countermeasures,” Special Rep., 2011.
  - [14] U. Guin, D. DiMase, and M. Tehranipoor, “A comprehensive framework for counterfeit defect coverage analysis and detection assessment,” *J. Electron. Testing*, vol. 30, no. 1, pp. 25–40, 2014.
  - [15] U. Guin and M. Tehranipoor, “On selection of counterfeit IC detection methods,” in *Proc. IEEE North Atlantic Test Workshop*, May 2013.
  - [16] K. Huang, J. Carulli, and Y. Makris, “Parametric counterfeit IC detection via support vector machines,” in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, 2012, pp. 7–12.
  - [17] Y. Jin and Y. Makris, “Hardware Trojan detection using path delay fingerprint,” in *Proc. IEEE Int. Workshop Hardware-Oriented Security Trust*, 2008, pp. 51–57.
  - [18] X. Zhang, K. Xiao, and M. Tehranipoor, “Path-delay fingerprinting for identification of recovered ICs,” in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, 2012, pp. 13–18.
  - [19] G. Contreras, T. Rahman, and M. Tehranipoor, “Secure split-test for preventing IC piracy by untrusted foundry and assembly,” in *Proc. Int. Symp. Defect Fault Tolerance VLSI Syst.*, 2013, pp. 196–203.
  - [20] Y. M. Alkabani and F. Koushanfar, “Active hardware metering for intellectual property protection and security,” in *Proc. USENIX Security Symp.*, Berkeley, CA, USA, 2007, pp. 20:1–20:16.
  - [21] F. Koushanfar and G. Qu, “Hardware metering,” in *Proc. Design Autom. Conf.*, 2001, pp. 490–493.
  - [22] K. Chatterjee and D. Das, “Semiconductor manufacturers’ efforts to improve trust in the electronic part supply chain,” *IEEE Trans. Compon. Packag. Technol.*, vol. 30, no. 3, pp. 547–549, Sep. 2007.
  - [23] V. V. de Leest and P. Tuyls, “Anti-counterfeiting with hardware intrinsic security,” in *Proc. Design Autom. Test Eur. Conf.*, 2013, pp. 1137–1142.
  - [24] R. Pappu, “Physical one-way functions,” Ph.D. dissertation, Program Media Arts Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 2001.
  - [25] U. Guin, X. Zhang, D. Forte, and M. Tehranipoor, “Low-cost on-chip structures for combating die and IC recycling,” in *Proc. ACM/IEEE Design Autom. Conf.*, 2014, DOI: 10.1145/2593069.2593157.
  - [26] X. Zhang, N. Tuzzio, and M. Tehranipoor, “Identification of recovered ICs using fingerprints from a light-weight on-chip sensor,” in *Proc. IEEE Design Autom. Conf.*, Jun. 2012, pp. 703–708.
  - [27] X. Zhang and M. Tehranipoor, “Design of on-chip lightweight sensors for effective detection of recycled ICs,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 5, pp. 1016–1029, May 2014.
  - [28] M. Miller, J. Meraglia, and J. Hayward, “Traceability in the age of globalization: A proposal for a marking protocol to assure authenticity of electronic parts,” in *Proc. SAE Aerosp. Electron. Avion. Syst. Conf.*, Oct. 2012, DOI: 10.4271/2012-01-2104.
  - [29] C. Kuemin, L. Nowack, L. Bozano, N. D. Spencer, and H. Wolf, “Oriented assembly of gold nanorods on the single-particle level,” *Adv. Funct. Mater.*, vol. 22, no. 4, pp. 702–708, 2012.
  - [30] U. Guin, D. Forte, and M. Tehranipoor, “Anti-counterfeit techniques: From design to resign,” in *Proc. Microprocessor Test Verif.*, 2013, to be published.
  - [31] L. W. Kessler and T. Sharpe, “Faked parts detection,” 2010. [Online]. Available: <http://www.circuitsassembly.com/cms/component/content/article/159/9937-smt>
  - [32] M. Tehranipoor and F. Koushanfar, “A survey of hardware Trojan taxonomy and detection,” *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 10–25, Jan./Feb. 2010.
  - [33] SAE International, “Counterfeit electronic parts; Avoidance, detection, mitigation, and disposition,” 2009. [Online]. Available: <http://standards.sae.org/as5553/>
  - [34] SAE International, “Fraudulent/counterfeit electronic parts; Tool for risk assessment of distributors,” 2011. [Online]. Available: <http://standards.sae.org/arp6178/>
  - [35] SAE International, “Fraudulent/counterfeit electronic parts; Avoidance, detection, mitigation, and disposition—Distributors counterfeit electronic parts; Avoidance protocol, distributors,” 2012. [Online]. Available: <http://standards.sae.org/as6081/>
  - [36] SAE International, “Fraudulent/counterfeit electronic parts; Avoidance, detection, mitigation, and disposition—authorized/franchised distribution,” [Online]. Available: <http://standards.sae.org/wip/as6496/>
  - [37] SAE International, “Test methods standard; Counterfeit electronic parts,” [Online]. Available: <http://standards.sae.org/wip/as6171/>
  - [38] Components Technology Institute (CTI), “Certification for counterfeit components avoidance program,” Sep. 2011. [Online]. Available: <http://www.cti-us.com/pdf/CCAP101Certification.pdf>
  - [39] Independent Distributors of Electronics Association (IDEA), “Acceptability of electronic components distributed in the open market,” [Online]. Available: <http://www.idofea.org/products/118-idea-std-1010b>
  - [40] J. Goldstein et al., *Scanning Electron Microscopy and X-ray Microanalysis*. New York, NY, USA: Springer-Verlag, 2003.
  - [41] B. Hu and M. Nuss, “Imaging with terahertz waves,” *Opt. Lett.*, vol. 20, no. 16, pp. 1716–1718, 1995.
  - [42] H. C. Chou, J. Zeller, U. Guin, M. Tehranipoor, and M. Anwar, “Time-domain THz spectroscopy for counterfeit IC detection,” in *Terahertz Physics, Devices, and Systems VII: Advanced Applications in Industry and Defense*, Apr. 2013.
  - [43] G. F. Nelson and W. F. Boggs, “Parametric tests meet the challenge of high-density ICs,” *Electronics*, vol. 48, no. 5, pp. 108–111, Dec. 1975.
  - [44] M. Soma, “Fault coverage of dc parametric tests for embedded analog amplifiers,” in *Proc. Int. Test Conf.*, Oct. 1993, pp. 566–573.
  - [45] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. New York, NY, USA: Springer-Verlag, Nov. 2000.
  - [46] P. Mazumder and K. Chakraborty, *Testing and Testable Design of High-Density Random-Access Memories*. New York, NY, USA: Springer-Verlag, Sep. 1996.
  - [47] U.S. Department of Defense, “Test method standard: Microcircuits,” 2010. [Online]. Available: <http://www.landandmaritime.dla.mil/Downloads/MilSpec/Docs/MIL-STD-883/std883.pdf>
  - [48] U.S. Department of Defense, “Test method standard: Test methods for semiconductor devices,” 2012. [Online]. Available: <http://www.landandmaritime.dla.mil/Downloads/MilSpec/Docs/MIL-STD-750/std750.pdf>
  - [49] R. D. Eldred, “Test routines based on symbolic logical statements,” *J. Assoc. Comput. Mach.*, vol. 6, no. 1, pp. 33–37, Jan. 1959.
  - [50] J. M. Galey, R. E. Norby, and J. P. Roth, “Techniques for the diagnosis of switching circuit failures,” in *Proc. 2nd Annu. Symp. Switching Circuit Theory Logical Design*, Oct. 1961, pp. 152–160.
  - [51] S. Seshu and D. N. Freeman, “The diagnosis of asynchronous sequential switching systems,” *IRE Trans. Electron. Comput.*, vol. EC-11, no. 4, pp. 459–465, Aug. 1962.
  - [52] Retronix, “Retronix curve trace test capability,” [Online]. Available: <http://www.retronix.com/uploads/downloads/Curve>
  - [53] Kozio, Inc., “A functional test approach for counterfeit, substandard, and high risk microcircuit detection,” [Online]. Available: <http://www.eejournal.com/archives/on-demand/2013021901-kozio/>
  - [54] H.-G. Stratigopoulos and Y. Makris, “Error moderation in low-cost machine-learning-based analog/RF testing,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 2, pp. 339–351, Feb. 2008.
  - [55] I. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer-Verlag, 1986.
  - [56] Synopsys, “HSPICE user guide,” 2010.
  - [57] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, “Trojan detection using IC fingerprinting,” in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 296–310.
  - [58] K. Hu, A. Nowroz, S. Reda, and F. Koushanfar, “High-sensitivity hardware Trojan detection using multimodal characterization,” in *Proc. Design Autom. Test Eur. Conf.*, 2013, pp. 1271–1276.
  - [59] X. Wang, H. Salmani, M. Tehranipoor, and J. Plusquellic, “Hardware Trojan detection and isolation using current integration and localized current analysis,” in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, 2008, pp. 87–95.
  - [60] R. Rad, X. Wang, M. Tehranipoor, and J. Plusquellic, “Power supply signal calibration techniques for improving detection resolution to hardware Trojans,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2008, pp. 632–639.
  - [61] C. Mouli and W. Carriker, “Future FAB: How software is helping Intel go nano- and beyond,” *IEEE Spectrum*, vol. 44, no. 3, pp. 38–43, Mar. 2007.
  - [62] F. Koushanfar, G. Qu, and M. Potkonjak, “Intellectual property metering,” in *Information Hiding*. New York, NY, USA: Springer-Verlag, 2001, pp. 81–95.
  - [63] G. Suh and S. Devadas, “Physical unclonable functions for device authentication and secret key generation,” in *Proc. Design Autom. Conf.*, 2007, pp. 9–14.
  - [64] S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, “Extended abstract: The butterfly PUF protecting IP on every FPGA,” in *Proc. IEEE Int. Workshop Hardware-Oriented Security Trust*, 2008, pp. 67–70.

- [65] K. Lofstrom, W. Daasch, and D. Taylor, "IC identification circuit using device mismatch," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2000, pp. 372–373.
- [66] J. Lee et al., "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Dig. Tech. Papers VLSI Circuits*, Jun. 2004, pp. 176–179.
- [67] Y. Su, J. Holleman, and B. Otis, "A 1.6 pJ/bit 96% stable chip-ID generating circuit using process variations," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2007, pp. 406–411.
- [68] F. Koushanfar, "Integrated circuits metering for piracy protection and digital rights management: An overview," in *Proc. Great Lakes Symp. VLSI*, 2011, pp. 449–454.
- [69] R. Chakraborty and S. Bhunia, "Hardware protection and authentication through netlist level obfuscation," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2008, pp. 674–677.
- [70] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICS for piracy prevention and digital right management," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2007, pp. 674–677.
- [71] J. Roy, F. Koushanfar, and I. Markov, "Epic: Ending piracy of integrated circuits," in *Proc. Design Autom. Test Eur.*, Mar. 2008, pp. 1069–1074.
- [72] J. Huang and J. Lach, "IC activation and user authentication for security-sensitive systems," in *Proc. IEEE Int. Workshop Hardware-Oriented Security Trust*, Jun. 2008, pp. 76–80.
- [73] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 66–75, Jan./Feb. 2010.
- [74] R. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1493–1502, Oct. 2009.
- [75] J. Rajendran, O. Sinanoglu, R. Karri, and Y. Pino, "Logic encryption: A fault analysis perspective," in *Proc. Design Autom. Test Eur. Conf.*, 2012, pp. 953–958.
- [76] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis for logic obfuscation," in *Proc. Design Autom. Conf.*, 2012, pp. 83–89.
- [77] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust*. New York, NY, USA: Springer-Verlag, 2012.
- [78] B. Barak et al., "On the (im)possibility of obfuscating programs," in *Proc. 21st Annu. Int. Cryptol. Conf.*, 2001, pp. 1–18.
- [79] R. Jarvis and M. G. McIntyre, "Split manufacturing method for advanced semiconductor circuits," U.S. Patent 7 195 931, 2004.
- [80] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Proc. Design Autom. Test Eur. Conf.*, 2013, pp. 1259–1264.
- [81] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proc. ACM Conf. Comput. Commun. Security*, 2013, pp. 709–720.
- [82] J. Rajendran, O. Sinanoglu, and R. Karri, "VLSI testing based security metric for IC camouflaging," in *Proc. IEEE Int. Test Conf.*, 2013, DOI: 10.1109/TEST.2013.6651879.
- [83] E. Charbon, "Hierarchical watermarking in IC design," in *Proc. IEEE Custom Integr. Circuits Conf.*, May 1998, pp. 295–298.
- [84] A. Kahng et al., "Constraint-based watermarking techniques for design IP protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 10, pp. 1236–1252, Oct. 2001.
- [85] E. Castillo, U. Meyer-Baese, A. Garcia, L. Parrilla, and A. Lloris, "Ipp@hdl: Efficient intellectual property protection scheme for IP cores," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 5, pp. 578–591, May 2007.
- [86] J. Lach, W. Mangione-Smith, and M. Potkonjak, "Fingerprinting techniques for field-programmable gate array intellectual property protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 10, pp. 1253–1261, Oct. 2001.
- [87] D. Kirovski, Y.-Y. Hwang, M. Potkonjak, and J. Cong, "Protecting combinational logic synthesis solutions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2687–2696, Dec. 2006.
- [88] B. Skoric, S. Maubach, T. Kevenaer, and P. Tuyls, "Information-theoretic analysis of capacitive physical unclonable functions," *J. Appl. Phys.*, vol. 100, no. 2, 2006, 024902.
- [89] P. Tuyls and L. Batina, "RFID-tags for anti-counterfeiting," in *Proc. Cryptographers' Track RSA Conf.*, 2006, pp. 115–131.
- [90] B. Gassend, D. Clarke, M. V. Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. Comput. Commun. Security Conf.*, 2002, pp. 148–160.
- [91] D. Lim et al., "Extracting secret keys from integrated circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 10, pp. 1200–1205, Oct. 2005.
- [92] S. Devadas et al., "Design and implementation of PUF-based 'unclonable' RFID ICs for anti-counterfeiting and security applications," in *Proc. IEEE Int. Conf. RFID*, 2008, pp. 58–64.
- [93] G. Suh, C. O'Donnell, and S. Devadas, "Aegis: A single-chip secure processor," *IEEE Design Test Comput.*, vol. 24, no. 6, pp. 570–580, Nov./Dec. 2007.
- [94] D. Holcomb, W. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.
- [95] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Ruhrmair, "The bistable ring PUF: A new architecture for strong physical unclonable functions," in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2011, pp. 134–141.
- [96] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Ruhrmair, "Characterization of the ring PUF," in *Proc. Design Autom. Test Eur. Conf. Exhibit.*, 2012, pp. 1459–1462.
- [97] U.S. Defense Logistics Agency, "DNA authentication marking on items in FSC 5962," Aug. 2012. [Online]. Available: <https://www.dibbs.bsm.dla.mil/notices/msgdspl.aspx?msgid=685>
- [98] Center for Hardware Assurance, Security, and Engineering (CHASE), 2013. [Online]. Available: <http://www.chase.uconn.edu/arochase-special-workshop-on-counterfeit-electronics.php>
- [99] Semiconductor Industry Association (SIA), "Public comments—DNA authentication marking on items in FSC5962," Nov. 2012.

## ABOUT THE AUTHORS

**Ujjwal Guin** (Student Member, IEEE) received the B.E. degree from the Department of Electronics and Telecommunication Engineering, Bengal Engineering and Science University, West Bengal, India, in 2004 and the M.Sc. degree from the Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA, USA, in 2010. Currently, he is working toward the Ph.D. degree at the Electrical and Computer Engineering Department, University of Connecticut, Storrs, CT, USA.

His current research interests include counterfeit detection and avoidance, hardware security, very large scale integration (VLSI) testing, and reliability. He is an active participant in SAE International's G-19A Test Laboratory Standards Development Committee.

Mr. Guin received the Best Student Paper Award at the 2013 North Atlantic Test Workshop (NATW). He was awarded the SIGDA Ph.D. Forum scholarship at the 2014 Design Automation Conference (DAC).



**Ke Huang** (Member, IEEE) received the M.S. degree in electrical engineering from the Joseph Fourier University (Grenoble I University), Grenoble, France, in 2008 and the Ph.D. degree in electrical engineering from the University of Grenoble, Grenoble, France, in 2011.

After spending two years as a Postdoctoral Research Associate at the University of Texas at Dallas, Richardson, TX, USA, he joined San Diego State University, San Diego, CA, USA, as an Assistant Professor in 2014. His research focuses on applications of data mining and machine learning in hardware security, reliability, and analog/radio-frequency (RF) integrated circuit (IC) testing.

Dr. Huang was awarded a Ph.D. Fellowship from the French Ministry of National Education from 2008 to 2011. He was a recipient of the Second Place Winner Award at the IEEE Computer Society Test Technology Technical Council (TTTC) E.J. McCluskey doctoral thesis competition in 2013 and a recipient of the Best Paper Award from the 2013 Design Automation and Test in Europe (DATE'13) conference.



**Daniel DiMase** received a B.S. degree in electrical engineering from University of Rhode Island, Kingston, RI, in 1989, and an Executive M.B.A. from Northeastern University, Boston, MA, in 2010.

He is currently the Director of Compliance and Quality at Honeywell International Inc. (Providence, RI), working in the counterfeit parts prevention team for the Aerospace strategic business group. He has over 20 years of industry experience, previously serving in leadership positions as president of SemiXchange, Inc. and ERAI. His work and research includes mitigating counterfeit and cyber risks, standards development and industry best practices and procedures for avoidance and detection of suspect counterfeit electronic parts and cyber vulnerabilities, supply-chain management, operations and finance, international logistics, global sourcing, risk management, and strategic planning.

Mr. DiMase is an active participant in SAE International's G-19 Counterfeit Electronic Parts Document Development group. He is chairman of the Test Laboratory Standards Development committee, co-chairman of the Distributor Process Rating committee, and actively participates on the Counterfeit Electronic Parts standard development committee for distributors. He is on the Department of Homeland Security's Customs and Border Protection Advisory Committee on Commercial Operations of CBP in the Intellectual Property Rights subcommittee. He received a special recognition award at the DMSMS and Standardization 2011 Conference for his leadership role in mitigating counterfeit parts. He has a Six-Sigma Green Certificate from Bryant University in 2008.



**John M. Carulli, Jr.** (Senior Member, IEEE) received the M.S.E.E. degree from the University of Vermont, Burlington, VT, USA, in 1990.

He is a Distinguished Member of the Technical Staff in the Analog Engineering Operations organization of Texas Instruments, Dallas, TX, USA. He was previously the manager of the product reliability and design reliability activities for new technology development in the External Development and Manufacturing division. His research interests include outlier analysis, product reliability modeling, performance modeling, and security.



**Mohammad Tehranipoor** (Senior Member, IEEE) received the M.Sc. degree in electrical engineering from University of Tehran, Iran, in 2000, and PhD degree in electrical and computer engineering from The University of Texas at Dallas, Richardson, TX, USA, in 2004.

He is currently the F.L. Castleman Associate Professor in Engineering Innovation at the University of Connecticut, Storrs, CT, USA. His current research projects include: computer-aided design and test for complementary metal-oxide-semiconductor (CMOS) very large scale integration (VLSI) designs, reliable systems design at nanoscale, counterfeit electronics detection and prevention, supply



chain risk management, and hardware security and trust. He has published over 200 journal articles and refereed conference papers and has given more than 110 invited talks and keynote addresses since 2006. He has published four books and ten book chapters.

Dr. Tehranipoor is a recipient of several best paper awards as well as the 2008 IEEE Computer Society (CS) Meritorious Service Award, the 2012 IEEE CS Outstanding Contribution, the 2009 National Science Foundation (NSF) CAREER Award, the 2009 UConn ECE Research Excellence Award, and the 2012 UConn SOE Outstanding Faculty Advisor Award. He serves on the program committee of more than a dozen of leading conferences and workshops. He served as Program Chair of the 2007 IEEE Defect-Based Testing (DBT) workshop, Program Chair of the 2008 IEEE Defect and Data Driven Testing (D3T) workshop, Co-program Chair of the 2008 International Symposium on Defect and Fault Tolerance in VLSI Systems (DFTS), General Chair for D3T-2009 and DFTS-2009, and Vice-General Chair for NATW-2011. He cofounded a new symposium called IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) and served as 2008 HOST and 2009 HOST General Chair and Chair of Steering Committee. He is currently serving as an Associate Editor-in-Chief (EIC) for IEEE DESIGN & TEST, an Associate Editor for Journal of Electronic Testing: Theory and Applications (JETTA), an Associate Editor for *Journal of Low Power Electronics*, an IEEE Distinguished Speaker, and an Association for Computing Machinery (ACM) Distinguished Speaker. He is a Senior Member of ACM and ACM SIGDA. He is currently serving as the director of the Center for Hardware Assurance, Security, and Engineering (CHASE) center.

**Yiorgos Makris** (Senior Member, IEEE) received the Diploma of Computer Engineering from the University of Patras, Patra, Greece, in 1995 and the M.S. and Ph.D. degrees in computer engineering from the University of California at San Diego, La Jolla, CA, USA, in 1998 and 2001, respectively.

After spending a decade on the faculty of Yale University, New Haven, CT, USA, he joined the University of Texas at Dallas, Richardson, TX, USA, where he is now a Professor of Electrical Engineering, leading the Trusted and RELiable Architectures (TRELA) Research Laboratory. His research focuses on applications of machine learning and statistical analysis in the development of trusted and reliable integrated circuits and systems, with particular emphasis in the analog/radio-frequency (RF) domain.

Prof. Makris served as the Program Chair of the IEEE VLSI Test Symposium in 2013–2014 and of the Test Technology Educational Program (TTEP) in 2010–2012. He also served as a guest editor for the IEEE TRANSACTIONS ON COMPUTERS and as a topic coordinator and/or program committee member for several IEEE and ACM conferences. He is a recipient of the 2006 Sheffield Distinguished Teaching Award and a recipient of the Best Paper Award from the 2013 Design Automation and Test in Europe (DATE'13) conference. His research activities have been supported by the National Science Foundation (NSF), the U.S. Army Research Office (ARO), Semiconductor Research Corporation (SRC), the Defense Advanced Research Projects Agency (DARPA), Boeing, IBM, LSI, Intel, and Texas Instruments.



# Hardware Trojan Attacks: Threat Analysis and Countermeasures

*This paper is a survey of the state-of-the-art Trojan attacks, modeling, and countermeasures.*

By SWARUP BHUNIA, *Senior Member IEEE*, MICHAEL S. HSIAO, *Fellow IEEE*, MAINAK BANGA, *Member IEEE*, AND SEETHARAM NARASIMHAN

**ABSTRACT** | Security of a computer system has been traditionally related to the security of the software or the information being processed. The underlying hardware used for information processing has been considered trusted. The emergence of hardware Trojan attacks violates this root of trust. These attacks, in the form of malicious modifications of electronic hardware at different stages of its life cycle, pose major security concerns in the electronics industry. An adversary can mount such an attack with an objective to cause operational failure or to leak secret information from inside a chip—e.g., the key in a cryptographic chip, during field operation. Global economic trend that encourages increased reliance on untrusted entities in the hardware design and fabrication process is rapidly enhancing the vulnerability to such attacks. In this paper, we analyze the threat of hardware Trojan attacks; present attack models, types, and scenarios; discuss different forms of protection approaches, both proactive and reactive; and describe emerging attack modes, defenses, and future research pathways.

**KEYWORDS** | Hardware intellectual property (IP) trust; hardware obfuscation; hardware Trojan attacks; self-referencing; side-channel analysis; Trojan detection; Trojan taxonomy; Trojan tolerance

Manuscript received October 30, 2013; revised May 20, 2014; accepted June 24, 2014. Date of publication July 15, 2014; date of current version July 18, 2014. This research was supported in part by the National Science Foundation (NSF) under Grants CNS-1054744 and DUE-1245756.

**S. Bhunia** is with the Case Western Reserve University, Cleveland, OH 44106 USA (e-mail: skb21@case.edu).

**M. S. Hsiao** is with the Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 USA (e-mail: hsiao@vt.edu).

**M. Banga** is with Intel Corporation, Folsom, CA 95630 USA (e-mail: mainak.banga@intel.com).

**S. Narasimhan** is with Intel Corporation, Hillsboro, OR 97124 USA (e-mail: seetharam.narasimhan@intel.com).

Digital Object Identifier: 10.1109/JPROC.2014.2334493

## I. THE THREAT

Hardware Trojan attacks have emerged as a major security concern for integrated circuits (ICs) [1]–[5]. These attacks relate to malicious modifications of an IC during design or fabrication in an untrusted design house or foundry, which involve untrusted people, design tools, or components. Such modifications can give rise to undesired functional behavior of an IC, or provide covert channels or backdoor through which sensitive information can be leaked. An adversary is expected to make a Trojan stealthy in nature that evades detection through conventional postmanufacturing test, but manifests during long hours of field operation. For an IC with moderate complexity, the number of possible Trojans can be inordinately large with varying activation mechanisms (referred to as triggers) and effects (referred to as payloads). Fig. 1 shows a simplified block diagram of a hardware Trojan, which causes a malfunction (by modifying signal  $S$  to  $S'$ ), when triggered—i.e., when the activation condition realized by the trigger logic is true. Such malicious inclusions effectively act as “spies or terrorists” on chip and can be extremely powerful, potentially leading to catastrophic consequences in diverse applications [15].

These malicious circuits have been popularly referred to as “hardware Trojans” similar to the software Trojans, which attack the operating system (OS) of a computer. The nomenclature is derived from a mythological incident attributed to the ancient Greeks in the Trojan war, where a wooden horse was gifted to the Trojan army who took it into their city walls without realizing that the enemy (Greek) soldiers were hidden inside the hollow horse. The seemingly trustworthy horse conditionally turned into a powerful and malicious weapon that drastically affected the course of the Trojan war. Similar to its mythical analogy, the two main features of a hardware Trojan are as

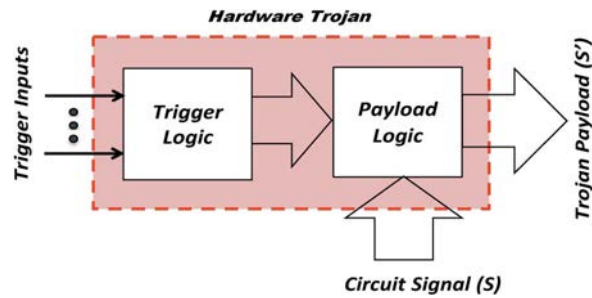


**Fig. 1.** General structure of a hardware Trojan in a design.

follows: 1) it should have a malicious intent; and 2) it should evade detection under conventional postmanufacturing test/validation process.

Current economic trend plays a major role in enhancing the vulnerability to Trojan attacks [1]–[3]. IC design and manufacturing practices increasingly rely on untrusted parties and entities in the IC life cycle. Economic factors dictate that most of the modern ICs are manufactured in unsecured fabrication facilities. Moreover, modern IC design often involves intellectual property (IP) cores supplied by untrusted third-party vendors, outsourced design and test services, as well as electronic design automation (EDA) software tools supplied by different vendors. Such a business model has, to a large extent, relinquished the control that IC design houses had over the design and manufacture of ICs making them vulnerable to malicious implants. Fig. 2 illustrates different steps of a typical IC life cycle and the possibility of Trojan attacks in these steps [3]. Each party associated with the design and fabrication of an IC can be a potential adversary who can tamper it [5]. Such tampering can be accomplished through

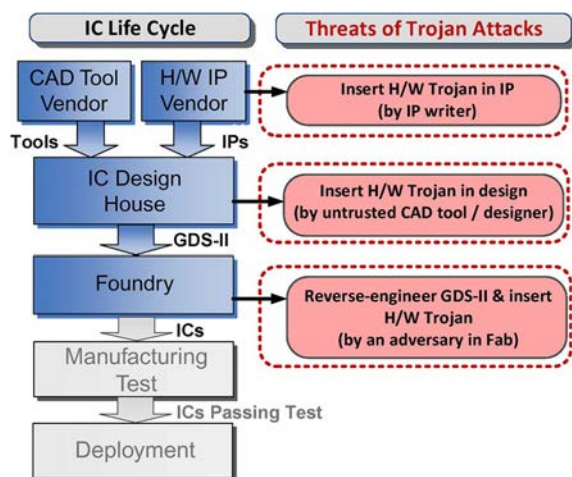


**Fig. 2.** Hardware Trojan attacks by different parties at different stages of IC life cycle.

add/delete/alteration of circuit structure or through modification of manufacturing process steps that causes reliability issues in ICs. From an attacker's perspective, the objective of such attacks can be manifold, e.g., to malign the image of a company to gain competitive edge in the market; disrupt major national infrastructure by causing malfunction in electronics used in mission-critical systems; or leak secret information from inside a chip to illegally access a secure system. Concerns about this vulnerability and the resultant compromise of system security have been expressed globally [3], [10], [11], especially since recent discoveries point to feasibility of such attacks [12]–[14]. Moreover, several unexplained military mishaps in the past have been attributed to the presence of malicious hardware modifications [1], [35]. Recent investigations have shown that an intelligent adversary can mount a hard-to-detect Trojan attack using just a few transistors or logic gates in a large multimillion transistor system-on-chip (SoC) design, or by selectively changing specific process steps, e.g., the doping profile, to affect the operational reliability of a circuit [14].

Ideally, any undesired modification made to an IC should be detectable by pre-silicon verification/simulation or post-silicon testing. However, pre-silicon verification or simulation requires a completely specified golden model of the entire IC. This might not be always available, especially for IP-based designs where IPs can come from third-party vendors. Besides, a large multimodule design is usually not amenable to exhaustive verification [6]. Post-silicon, the design can be verified either through destructive depackaging and reverse engineering of the IC [3], or by comparing its functionality or circuit characteristics with a golden version of the IC [4], [7], [8]. However, state-of-the-art approaches do not allow destructive verification of ICs to be either cost effective or scalable [6], [9]. Moreover, as pointed out in [3], it is possible for the adversary to insert Trojans in only some ICs on a wafer, not the entire population, which limits the usefulness of a destructive approach. Traditional postmanufacturing testing is not suitable for detecting hardware Trojans. This is due to the stealthy nature of hardware Trojans and the vast spectrum of possible Trojan instances an adversary can exploit.

Hardware Trojan attacks raise a new set of challenges for trusted operation of electronics in field [15]. It demands trust validation of ICs with respect to malicious design modification at various stages of the IC lifecycle, where untrusted components/personnel are involved. In particular, it introduces the requirement for reliable detection of any malicious design modification during postmanufacturing test. It also imposes a requirement for trust validation in hardware IP cores obtained from untrusted third-party vendors.

### A. Trojan Versus Fault

Postmanufacturing test, both structural and functional, is targeted to detect different types of manufacturing

**Table 1** Comparison Between Faults and Hardware Trojan Attacks

|                        | Fault                                                            | Hardware Trojan                                                               |
|------------------------|------------------------------------------------------------------|-------------------------------------------------------------------------------|
| <b>Activation</b>      | Usually at known functional state                                | Arbitrary combination/sequence of internal circuit states (digital/analog)    |
| <b>Insertion Agent</b> | <i>Accidental</i> (due to imperfection in manufacturing process) | <i>Intentional</i> (inserted by an adversary during IC design or fabrication) |
| <b>Manifestation</b>   | Functional/parametric failure                                    | Functional/parametric failure or information leakage                          |

defects. Faults such as stuck-at-faults or path delay faults are logical models of physical defects (e.g., resistive short or open). Table 1 compares the properties of hardware Trojans with those of faults. Unlike faults in a design, which occur due to imperfections introduced during the manufacturing process, hardware Trojans are deliberately inserted by an adversary to serve a specific malicious purpose. While a fault is activated at a known functional state of a circuit (e.g., a stuck-at-0 fault at the output of an inverter can be activated by sensitizing its input to a logical value of 0), a Trojan can be designed to activate at an arbitrary complex condition, including a sequence of events at internal circuit nodes.

### B. Software Versus Hardware Trojans

Table 2 distinguishes hardware Trojans from its software counterparts in terms of key properties. A software Trojan horse (STH) is a type of malware program with malicious code that gains privileged access to the OS and may steal information or cause harm to the host computer (e.g., erase or corrupt data) [16], [67]. An STH attack can often be cured in the field by running an anti-Trojan program (such as the one in [76]) that monitors and removes the Trojan [68], [69]. Hardware Trojans are inserted into an IC before it is fabricated. Unlike an STH, a hardware Trojan is virtually impossible to remove from a chip after fabrication and hence can be extremely difficult to remedy during field operation.

### C. Trojan Attacks Through Hardware IP or CAD Tools

SoC design based on reusable hardware IP is now a pervasive practice in the semiconductor industry due to the dramatic reduction in design/verification cost and time it offers. This growing reliance on reusable preverified hardware IPs and a wide array of design automation tools

during SoC design—often gathered from untrusted third-party vendors—severely affects the security and trustworthiness of SoC computing platforms [17]. The possibility of Trojan attacks in third-party IP poses a major integrity concern to SoC designers. Verification of trust of an IP acquired from untrusted third-party sources can be extremely challenging due to lack of golden models. Conventional verification approaches, which rely on the presence of a golden or reference design, do not work in case of third-party IPs. On the other hand, functional simulation or emulation does not provide adequate coverage due to often incomplete functional specifications. Thus, even though simulation can validate the correctness of a design with respect to functional specifications, it cannot provide assurance against additional functionality due to a Trojan. For example, if a processor IP core triggers a malicious memory write for an “add” instruction with a specific rare combination of operand values, both simulation and emulation are very likely to fail, since they may not excite all rare conditions. Similarly, untrusted computer-aided design (CAD) tools can potentially cause malicious inclusions in a design [18]. Such attacks can be introduced early in the design cycle, so that they can evade subsequent verification steps. Often, a designer uses a CAD tool suite from the same vendor. In that case, a verification tool from a vendor can potentially ignore a malicious insertion by the synthesis engine from the same vendor.

### D. Complex Attack Mode

Malicious collusion between multiple parties at different stages of the design, manufacturing, and deployment can make a hardware Trojan attack more potent. One such example is presented in [19], where the inserted Trojan circuitry leaks information through a covert side channel that allows conspiring malicious parties to discover the encryption key. Another example is presented in [20], where focused ion beams help insert a dormant Trojan to a

**Table 2** Software Versus Hardware Trojan Attacks

|                   | Software Trojan                                                                    | Hardware Trojan                                                    |
|-------------------|------------------------------------------------------------------------------------|--------------------------------------------------------------------|
| <b>Activation</b> | A type of malware that resides in a code and activates during its execution        | Resides in hardware (e.g. IC) and activates during its operation   |
| <b>Infection</b>  | Spreads through user interaction e.g. downloading and running a file from Internet | Inserted through untrusted entities in design or fabrication house |
| <b>Remedy</b>     | Can be <i>removed in field</i> through S/W support                                 | <i>Cannot be removed</i> once IC is fabricated                     |

functional unit in the fabrication facility. A general model of such an attack, referred to as multilevel attack, is described in [21]. Using a crypto module as a case study, it analytically shows that the resultant attack poses a significantly stronger threat than that from a single adversary. Such attacks can easily bypass both pre-silicon verification as well as postmanufacturing test. An example would be a fault attack in an advanced encryption system (AES) module, which enables only the malicious parties, who are part of the conspiracy, to retrieve the cipher key. The Trojan could also be a passive hardware entity, which simply serves to help malicious software circumvent the hardware protection mechanisms. Moreover, a Trojan circuit can be localized or distributed in a chip, and a trigger condition or the payload of a Trojan can be digital or analog, e.g., a Trojan can be triggered by changes in temperature [15]. Becker *et al.* [60] have demonstrated that by selectively changing the dopant level in the

transistors, they are able to create Trojans that are resistant against existing detection techniques. These Trojans are nonintrusive and difficult to detect since they do not alter any functionality and are not optically observable.

## II. TROJAN MODEL, INSTANCES, AND CLASSIFICATION

### A. Trojan Models and Examples

An intelligent adversary is expected to hide such tampering with an IC's behavior in a way that makes it extremely difficult to detect with conventional postmanufacturing testing [5]. Intuitively, it means that the adversary would ensure that such tampering is manifested or triggered under very rare conditions at the internal nodes, which are unlikely to arise during testing but can occur during long hours of field operation [22], [23]. Fig. 3

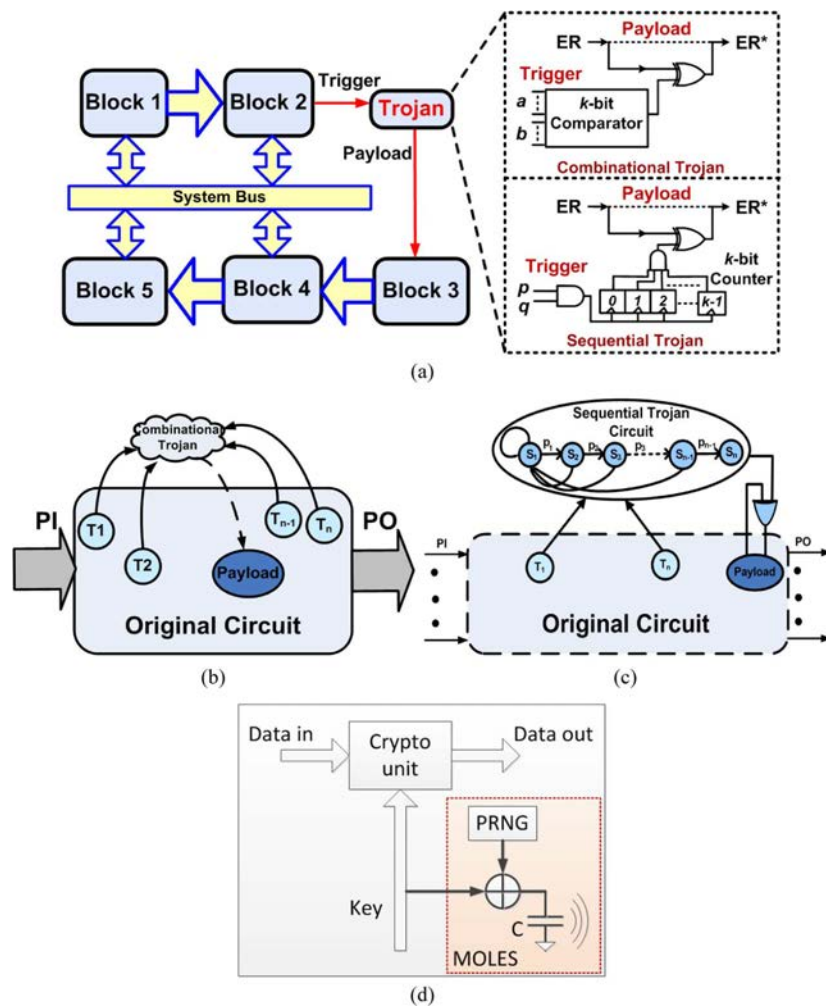


**Fig. 3.** Hardware Trojan attacks in different forms: (a) combinational and sequential Trojans; general model of (b) combinational and (c) sequential Trojan, which can facilitate test generation for Trojan detection; (d) Trojans with capability of leaking secret information from inside a crypto chip through power side channels [19].

shows some examples of different types of hardware Trojans. The combinational Trojan, as shown in Fig. 3(a), does not contain any state element (e.g., flip-flop or latch) and depends only on the simultaneous occurrence of a set of rare node conditions (e.g., a predefined value on node sets  $a$  and  $b$ ) to trigger a malfunction (by flipping signal  $ER$ ). The sequential Trojan shown in Fig. 3(a), on the other hand, undergoes a sequence of state transitions before triggering a malfunction. Fig. 3(a) shows a synchronous  $k$ -bit counter, which activates when the count reaches  $2^k - 1$ , by modifying the node  $ER$  to an incorrect value at node  $ER^*$ . Fig. 3(b) and (c) show general models of combinational and sequential Trojans, respectively. These abstract models of Trojans are useful for studying the space of possible Trojans, and, similar to fault models, help in test vector generation for Trojan detection.

1) *Trojans in Cryptographic Engines*: A possible Trojan attack in a crypto engine can try to subvert the security mechanisms. The payload could range from a mechanism that presents dummy keys, predefined by the attacker, instead of the actual cryptographic keys used for sensitive encryption or signature verification operations, to leaking the secret hardware keys via covert side channels, e.g., information leaked through a power trace. Fig. 3(d) provides an example of such a Trojan that attempts to leak a secret key from inside a cryptographic IC through power side-channels using a technique called malicious off-chip leakage enabled by side channels (MOLES) [19]. Even if the IC has been designed to minimize side-channel information leakage, a hardware modification could help overcome the protection under specific circumstances where the attacker is in possession of the system or physically near the system to extract the secret information. Other targets could be a random number generator used for deriving random session keys for a particular operation or the debug passwords used for unlocking test-

mode access to security-sensitive signals. Researchers have also proposed leaking such secret information over wireless channels [77] by using low-bandwidth modulation of the transmitted signal.

2) *Trojans in General-Purpose Processors*: In general-purpose processors, an attacker at the fabrication facility can implement a backdoor, which can be exploited in the field by a software adversary [23]–[26]. For example, modern processors implement a hardware chain of trust to ensure that malware cannot compromise the hardware assets such as secret keys and memory range protections. By using different stages of firmware and boot code authentication, one can ensure that the operating system (OS) kernel and lower levels (such as hypervisor) are not corrupted. However, in such systems, the attacker at an untrusted fabrication facility could implement a backdoor which disables the secure booting mechanism under certain rare conditions or when presented with a unique rare input condition in the hands of an end-user adversary [23]. Similarly, other objectives which could be realized with the help of hardware Trojans would be to bypass memory range protections using buffer overflow attacks or to gain access to privileged assets by evading the access control protection mechanisms implemented in the hardware.

## B. Trojan Taxonomy

The taxonomy of Trojan circuits has been presented in various forms, and it continues to evolve as newer attacks and Trojan types are discovered. Here, we will present a high-level classification, as shown in Fig. 4, based on variations in activation mechanism and Trojan effect. Based on the trigger condition, the hardware Trojans can be classified into analog and digital Trojans. The former are activated by analog conditions such as temperature, delay, or device aging effect, whereas the latter are triggered by some Boolean logic function. Digitally triggered Trojans can again be classified

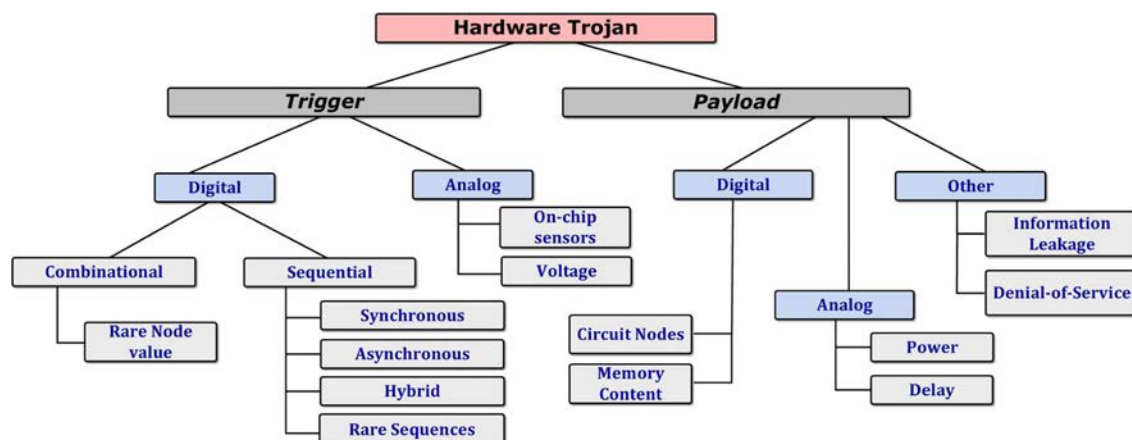


Fig. 4. Trojan taxonomy based on trigger and payload mechanisms.

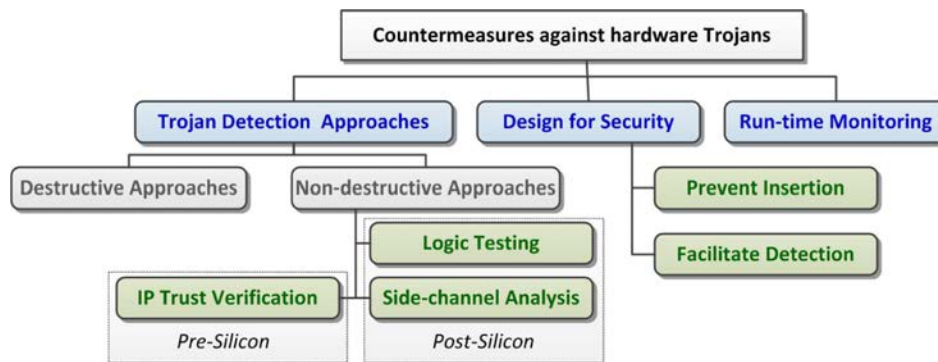


**Fig. 5.** Overview of different protection approaches against hardware Trojan attacks.

into combinational and sequential types. Analog Trojans include attacks on process steps that compromise reliability of all or select chips. These Trojans, referred to as reliability trojans by some researchers, may cause accelerated aging of the devices [27]. The reduction in reliability is caused by acceleration of the wearing out mechanisms for complementary metal–oxide–semiconductor (CMOS) transistors, such as negative bias temperature instability (NBTI) or hot carrier injection (HCI). Selective malicious changes in the manufacturing process, such as variation in nitrate concentration in the gate oxide layer or the temperature used during the nitrate layering process [27] can result in creation of infected ICs with a much shorter lifetime. What makes it challenging to detect such tampering is that it can be local to a particular functional circuit block or even to a small section of a block that can potentially evade standard postfabrication reliability characterization steps.

In terms of the payload, the Trojan can cause functional failure upon triggering or have a passive effect such as heating of the die or leaking of information. A Trojan can cause an “information leakage” attack, where secret information is leaked by a Trojan via a transmitted radio signal or serial data port interface such as the RS-232-C port. It could also involve a side-channel attack where the information is leaked through the power trace [19] or through thermal radiation or through optical modulation of an output light-emitting diode (LED) [28]. Another type of Trojan payload would be unauthorized alteration in system behavior, e.g., a denial-of-service (DoS) attack, which causes a system’s functionality to be unavailable.

Other parameters used to define and compare different Trojans include the hardware overhead and activation probability [5]. The area and power overhead relative to the design in which the Trojan is inserted has to be an undetectable fraction, in order to evade detection by obvious means [29]. One can reuse existing logic and unused states in existing finite state machines (FSMs) to reduce the overhead. Moreover, the layout of complex ICs typically contains unused space which can be used for inserting extra gates without affecting the die footprint.

The activation probability is also a tradeoff between avoiding detection and malicious impact.

### III. COUNTERMEASURES

#### A. Challenges

Conventional postmanufacturing tests using functional/structural/random patterns perform poorly to reliably detect hardware Trojans. This is because manufacturing test generation and application attempt to detect defects or unacceptable variations in device parameters that cause deviation from functional or parametric specifications. They do not identify additional functionalities due to a Trojan or deviation in circuit behavior triggered by arbitrary rare events. There are several important challenges with respect to reliable detection of Trojans using a post-silicon test/validation process. First, an adversary can exploit inordinately large number of Trojan instances of varying forms and sizes [30]. These Trojans vary widely in structural and functional properties including trigger conditions and payloads, which makes it difficult to develop a logical model of Trojans. The number of possible Trojan instances has a combinatorial dependence on the number of circuit nodes. As an example, even with the assumption of maximum of four trigger nodes and a single payload, a relatively small ISCAS-85 benchmark circuit c880 with 451 gates can have  $\sim 10^9$  triggers and  $\sim 10^{11}$  distinct Trojan instances, respectively [8]. Thus, it is not practical to enumerate all possible Trojan instances to generate test patterns or compute test coverage. Second, due to their stealthy nature, activating arbitrary Trojan instances and observing their effects can be extremely challenging. Hence, deterministic and exhaustive testing approaches appear infeasible. Third, with respect to observing Trojan effects in physical parameters, e.g., delay and supply current, process and measurement noise pose a major challenge. In particular, increasing process variations in advanced technologies can mask minuscule Trojan effects in a physical parameter.

## B. Class of Protections

Research efforts on protection against Trojan attacks have focused on three broad classes of solutions: 1) Trojan detection approaches; 2) design for security (DFS) approaches; and 3) runtime monitoring approaches. Fig. 5 shows a broad classification of the countermeasures against hardware Trojans. Trojan detection approaches typically attempt to detect the existence of Trojans at an IP level using pre-silicon techniques or using nondestructive techniques during post-silicon manufacturing test through a trust validation process. They can be further classified into logic testing approaches, which focus on generating appropriate test patterns for Trojan detection, and side-channel analysis approaches where a measurable parameter such as power, delay, temperature, or electromagnetic (EM) radiation profile can be used to isolate a Trojan-infected IC from the golden ones. The DFS approaches attempt to make insertion of hard-to-detect Trojans difficult or facilitate detection during post-silicon validation. However, the DFS or trust validation approaches usually are not capable of providing complete confidence against diverse forms of possible Trojan attacks. To safeguard against potentially undetected Trojans, runtime validation approaches can be employed based on online monitoring of circuit operation [15], [31]. Such approaches provide a last line of defense against Trojan attacks and often attempt to contain the effect of an activated Trojan (e.g., by entering a fail-safe mode).

## C. Why Destructive Reverse Engineering Is Not Effective

Once the fabrication and packaging of an IC is complete, we have limited visibility into the components inside it. One can, however, use destructive reverse-engineering techniques to depackage an IC and obtain microscopic images of each layer to reconstruct the design for trust validation of the end product. The destructive techniques [9], [32] use a sample of the manufactured ICs which are subject to demetallization using chemical mechanical polishing (CMP) followed by scanning electron microscope (SEM) image reconstruction and analysis [3]. Though it would take several weeks to months to do this for an IC of reasonable complexity, it has the potential of giving 100% assurance that any malicious modification in the IC will be detected. However, at the end of this invasive process, the IC cannot be used, and we only get

the information for a single IC. It is possible for the attacker to have infected only some samples in an entire lot of ICs by manipulating the layout masks. Hence, in general, destructive approaches are not considered viable for Trojan detection. However, destructive reverse engineering on a limited number of samples can be attractive in order to obtain the characteristics of a golden batch of ICs. This information can be useful for trust validation through side-channel analysis, as discussed in Section IV.

## IV. TROJAN DETECTION

Table 3 provides a comparison of advantages and disadvantages of two major Trojan detection paradigms: logic testing and side-channel analysis. Logic testing approaches, both functional and structural, attempt to develop directed test patterns to activate unknown Trojan instances and propagating their effects to output ports [7]. Although robust under process and measurement noise, these approaches are likely to fail to activate large Trojans consisting of large numbers of trigger inputs. An alternative approach is to measure a side-channel parameter, such as supply current or path delay, which can be affected due to unintended design modifications. However, the effectiveness of side-channel analysis is limited by large intrinsic device parameter variations in modern nanometer technologies. These detection approaches typically require a golden design or a set of golden ICs to compare the measured values in order to identify the Trojan-infected ones. Table 4 provides a comparison of the Trojan detection capability of alternative approaches. Logic testing and side-channel-analysis-based validation approaches provide complementary capabilities in detecting Trojans of different types and sizes. Hence, a postmanufacturing validation solution that combines the benefits of both approaches can be effective in maximizing the level of confidence. For applications, which require the highest level of trust against Trojan attack, one can combine postmanufacturing validation with online monitoring. Finally, validation approaches, both postmanufacturing and online, can be complemented with low-cost DFS solutions, which harden a design with respect to Trojan insertion or help in the validation process.

### A. Logic Testing

In order to detect the existence of a Trojan using logic testing, it is not only important to satisfy the trigger

**Table 3** Comparison of Logic Testing and Side-Channel-Analysis-Based Trojan Detection

|             | Logic Testing                                                                                                                         | Side-Channel Analysis                                                                                                               |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <b>Pros</b> | <ul style="list-style-type: none"> <li>• Robust under process noise</li> <li>• Effective for detecting ultra-small Trojans</li> </ul> | <ul style="list-style-type: none"> <li>• Effective for large Trojans</li> <li>• Easy to generate test vectors</li> </ul>            |
| <b>Cons</b> | <ul style="list-style-type: none"> <li>• Difficult to generate test vectors</li> <li>• Large Trojan detection challenging</li> </ul>  | <ul style="list-style-type: none"> <li>• Vulnerable to process noise</li> <li>• Ultra-small Trojan detection challenging</li> </ul> |

Table 4 Capability of Trojan Detection Schemes to Identify Different Hardware Trojan Types

| Trojan Size<br>Trojan Type |               | Large                                      |                              | Small                  |                     |
|----------------------------|---------------|--------------------------------------------|------------------------------|------------------------|---------------------|
|                            |               | Localized                                  | Distributed                  | Localized              | Distributed         |
| Digital                    | Combinational | Logic Testing                              | Side-Channel (IDDT) + Online | Logic Testing + Online | Side-Channel (IDDT) |
|                            | Sequential    | Side-Channel                               | Side-Channel (IDDQ)          | Side-Channel (IDDT)    | Side-Channel (IDDT) |
| Analog                     |               | Side-Channel (IDDT, IDDQ, Others) + Online |                              |                        |                     |

condition, but also to propagate the effect of such an event to an output node and observe it. Due to the inordinately large space of possible Trojans, as mentioned earlier, it is not practical to enumerate all possible Trojan instances to generate deterministic test patterns or compute test coverage. It indicates that, instead of an exact approach, a statistical approach for test vector generation can be computationally more tractable. This has motivated efforts to develop a statistical approach for Trojan detection. One such effort, referred to as multiple excitation of rare occurrence (MERO) is reported in [8]. The main objective of this methodology is to derive a set of test patterns that is compact (minimizing test time and cost), while maximizing the Trojan detection coverage. The basic concept is to detect low probability conditions at the internal nodes and then derive an optimal set of vectors that can trigger each of the selected low probability nodes individually to their rare logic values multiple times (e.g., at least  $N$  times, where  $N$  is a user-defined parameter). By increasing the toggling rate of nodes that are random-pattern resistant, it improves the probability of activating a Trojan compared to purely random patterns.

Fig. 6 illustrates the concept with two examples—one combinational and one sequential Trojan. The combinational Trojan in Fig. 6(a) is activated when  $a = 0$ ,  $b = 1$ , and  $c = 1$  are satisfied, while the sequential Trojan in

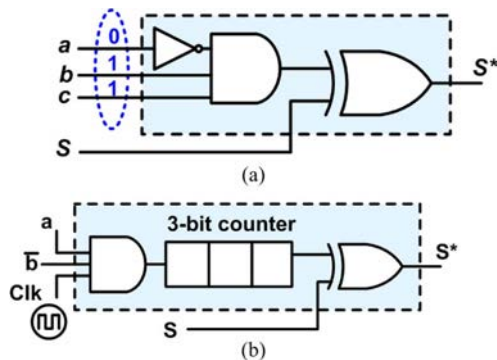


**Fig. 6.** Logic testing approach for hardware Trojan detection based on multiple excitation of rare conditions [8]: (a) for combinational Trojan; (b) for sequential Trojan.

Fig. 6(b) requires occurrences of  $a = 1$  and  $b = 0$  in several clock cycles to trigger. Hence, if we can generate a set of test vectors that induce these conditions at these nodes individually  $N$  times where  $N$  is sufficiently large (e.g.,  $> 100$ ), then a Trojan with a triggering condition composed jointly of these nodes is highly likely to be activated by the application of this test set. The MERO methodology is conceptually similar to the  $N$ -detect test used in stuck-at automatic test pattern generation (ATPG), where the test set is generated to detect each single stuck-at-fault in a circuit by at least  $N$  different patterns in order to improve test quality and defect coverage. Another approach of logic testing is to develop guided tests for detecting Trojan attacks in small but critical parts of a design [33]. For example, one can generate test vectors to observe undesired writes into a memory array or unjustifiable activity in the key-dependent logic of a crypto system. Waksman et al. [64] have used the stealthiness property of Trojan-affected nets to isolate them from a given design. Their tool, called FANCI, uses a scalable, Boolean functional analysis to detect these nets.

1) *Coverage Metric*: A metric similar to fault coverage, that provides a measure of confidence against arbitrary Trojan attacks, can be attractive for quantifying the effectiveness of a Trojan detection approach. Due to difficulties in enumerating all possible Trojan instances, it is infeasible to deterministically measure Trojan coverage. Statistical measures of confidence have been developed to address this issue. A random sampling approach can be used to compute both Trigger and Trojan coverage [8]. From the Trojan population, a representative set of Trojans with specific structure (e.g., number of trigger nodes) is randomly selected. From this set of sampled Trojans, ones with false trigger conditions that cannot be justified with any input pattern are eliminated. Then, a circuit under test is simulated for each vector in the given vector set and checked whether the triggering condition is satisfied. For an activated Trojan, if its effect can be observed at any primary output or scan flip-flop input, the Trojan is considered detected. The percentages of Trojans activated and detected constitute the trigger coverage and the Trojan coverage, respectively. To make such measures effective, however, it is important to consider an adequate number

of Trojans in the sample (as dictated by the principles of sampling theory) from the universal Trojan space.

### B. Side-Channel Analysis

In order to overcome the limitations of logic testing approaches, the use of side-channel analysis during post-silicon testing has been widely investigated [30], [34]–[42]. A malicious inclusion during design or fabrication is bound to have an effect on the power consumed by the circuit. In most cases, it is also likely to affect the delay of certain circuit paths. The impact on power is expected to be due to addition/alteration of circuit elements used in realizing the Trojan. It can cause deviation in the static (i.e., current drawn at idle state of a circuit) as well as in the transient (i.e., switching) current profile of the circuit under test. Most Trojans constantly monitor their activation condition. The act of monitoring consumes power, even if the change in power consumption is minimal; indeed that would be the goal of an adversary [4]. A delay impact can be due to either addition of extra logic level in a path (as in Fig. 6) or increase in capacitive load in a path. The impact of an unknown Trojan in physical parameters can be observed and compared between golden and Trojan-infected circuits to identify its existence.

Effectiveness of Trojan detection through side-channel analysis largely depends on two major parameters: 1) signal-to-noise ratio (SNR); and 2) the Trojan-to-circuit ratio (TCR). The signal of interest is the effect of a Trojan on a side-channel parameter. Noise is introduced by process and environmental variations. The effect of a Trojan on a side-channel parameter such as current or delay can be easily masked by the noise. Hence, simple comparison techniques can lead to a large number of false detections. This has led to a wide variety of postprocessing techniques to statistically isolate the Trojan effect from noise. On the other hand, the TCR indicates the gap between the original and Trojan-affected parameter. In an ideal scenario of zero noise, the detection sensitivity depends on the TCR. Hence, amplifying Trojan effect compared to the original value helps improving the confidence in identifying a Trojan instance. Zhang et al. [62] have proposed a virtual probe framework to get an accurate profiling of spatial variations within the chip without incurring much overhead cost. The problem is formulated as a maximum *a posteriori* (MAP) estimation problem and linear programming is used to solve it. Next, we present several side-channel analysis approaches based on different parameters such as transient current, static current, and path delay, and discuss their challenges and effectiveness.

1) *Static Current Analysis*: The presence of a Trojan circuit will be reflected in the current drawn from the power supply, even if no switching occurs in the Trojan circuit. Static CMOS gates are subject to leakage current in the idle mode, and this is more pronounced for submicrometer technologies. Any extra gate will consume

extra leakage power, which is additive and can ideally be used to distinguish golden circuits from ones with Trojan [34]–[36]. However, the leakage current effect due to a few extra gates often provides poor sensitivity when measuring the total leakage current for a multimillion gate SoC due to low TCR. Because of the exponential dependence of leakage current on transistor threshold voltage ( $V_T$ ) variations, chip-to-chip variations in leakage current can be very high, leading to low SNR. To increase sensitivity, one can measure current from multiple power pins, as shown in Fig. 7, thus effectively reducing the problem to that of detecting a few gates in a fraction of the total gates in the IC [34]. Since the leakage current of a logic gate is sensitive to the value at its inputs, a region-based approach using appropriate input vectors can also help localize a small region of the IC, which is infected by the Trojan.

Wei and Potkonjak [61] have employed a segmentation-based diagnosis approach to isolate Trojans in affected circuits. They use overall leakage current of the Trojan gates to identify them, even when they are not activated. The circuit under test is partitioned into a number of segments and the segments that dissipate higher leakage current are more closely inspected.

2) *Transient Current Analysis*: The goal of transient current ( $I_{DDT}$ ) analysis is to detect switching activity inside a Trojan circuit [30]. A Trojan detection approach based on this mechanism needs to carefully consider natural variations in current flow and minimize its influence. Fig. 8(a) illustrates device parameter variations, both chip to chip (interdie) and within chip (intradie), in a nanometer process. As observed from Fig. 8(a), variations in a device parameter such as  $V_T$  manifest themselves in variations at circuit level and mask the effect of Trojan in current and operating frequency ( $F_{max}$ ). Depending on the threshold, we can trade off the number of false positives

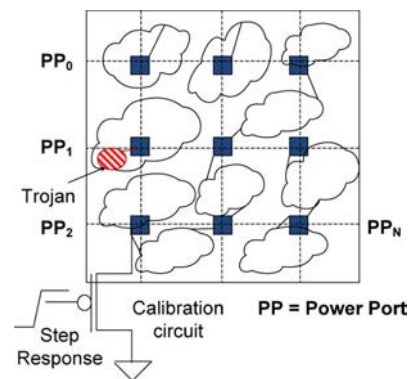


**Fig. 7. Region-based approach using current measurements from multiple power ports can be used for isolating Trojan effect with high sensitivity [40].**

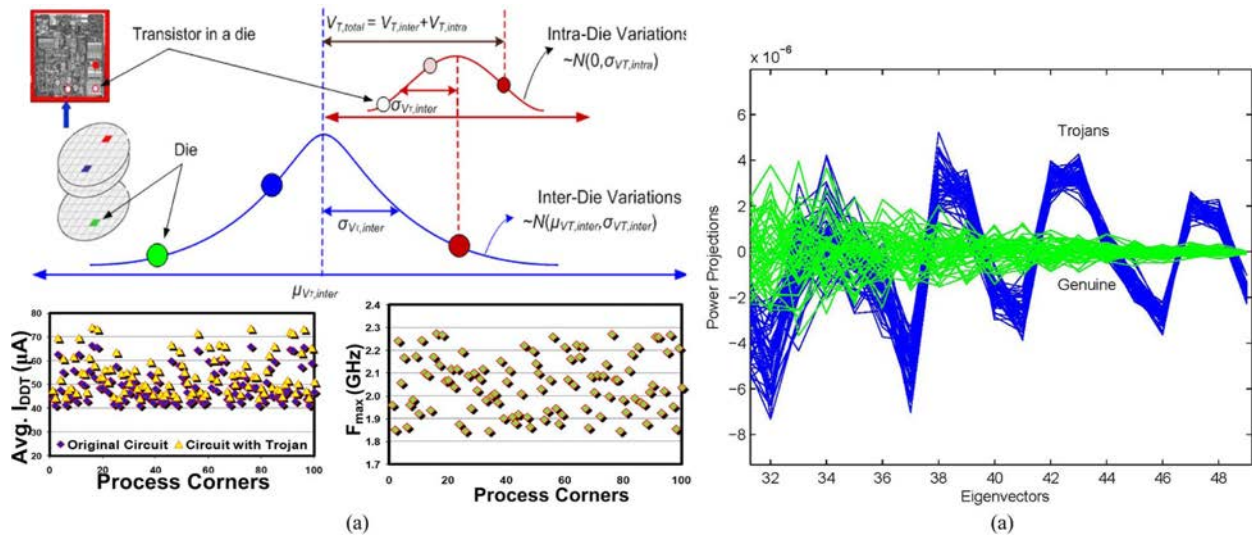


**Fig. 8. (a) Die-to-die and within-die variation in a device parameter (threshold voltage or  $V_t$ ) and corresponding impact in side-channel parameters: transient supply current ( $I_{DDT}$ ) and maximum frequency ( $F_{max}$ ) [30]. (b) Side-channel transient current signal from a Trojan can be separated and identified from the statistical distribution of process noise using Karhunen-Loeve expansion [4].**

(where we misclassify golden chips as Trojan-containing ICs) and false negatives (where we fail to identify Trojan ICs because they fall under the process noise margin). Similar variations can arise due to temporal effects such as voltage and temperature variations and measurement noise during testing.

The variability in the measurement process can be minimized by standardizing the measurement setup and averaging over multiple measurements to cancel out random noise. The choice of input vector set plays a significant role in improving TCR, thus improving detection sensitivity. Region-based partitioning [43] and directed test vector generation [33] to induce switching activity in possible Trojan instances can cause them to be easily detected. On the other hand, the sensitivity can be increased by decreasing the effect of process noise. Most of the existing approaches try to achieve that through process calibration by normalization and using statistical averaging techniques [37]–[40].

The IC fingerprinting technique [4] uses signal processing techniques such as the Karhunen-Loeve expansion to calibrate the process noise and identify subspaces from the transformed power trace signal, which highlight the presence of the Trojan, as shown in Fig. 8(b). From the analysis of power traces, it is possible to identify reasonably small Trojan instances, e.g., one with an equivalent area of 0.01% of the total size of the circuit, in the presence of  $\pm 7.5\%$  random parameter variations. Region-based approaches have been explored where careful selection of test vectors causes switching activity in a smaller functional region of the chip, thus decreasing the denominator of TCR equation. Such a divide-and-

conquer method can also take advantage of the different power pins of the chip having more sensitivity to different structural regions of a chip. By calibrating the power profile of these different pins, the effect of process noise can be reduced while highlighting the Trojan contribution to transient current. The interdie process variations can also be neutralized by comparing transient currents from different regions in a chip using a self-referencing matrix [42]. It also helps to localize a Trojan by pinpointing which region of an IC is infected with Trojan circuit. Such a diagnosis can be useful for tracing the source of a Trojan attack.

3) *Path Delay Analysis*: Another parameter of interest for Trojan detection is the path delay [38], [39]. Activating a Trojan may sensitize a functional path whose propagation delay is adversely affected by the malicious inclusion. However, impact of a Trojan in path delay can be small. In particular, for large path delays, a minor change in delay is likely to be masked by process variations. Hence, appropriate vector selection that leads to sensitization of paths affected by Trojans is of great importance. For sequential circuits without full scan, one needs to adopt design-time techniques for measuring all path delays, including short paths.

To isolate the Trojan effect from a side-channel parameter, the Trojan detection problem can be formulated as a linear programming problem (LPP) with process variation of each gate represented as a constant scaling factor for its leakage or delay under various input conditions [36]. This technique is capable of detecting a single extra gate in benchmark circuits. It can be further

refined with the effect of thermal variations on the leakage to break all correlations. Statistical measures are used to improve the detection accuracy and obtain a high degree of separation between golden and Trojan-affected instances. Like most other side-channel analysis approaches, however, the scalability of such an approach to large designs and its effectiveness in actual measurements require further study.

4) *Multiple-Parameter Analysis*: In order to reliably isolate the Trojan effect in the presence of process noise, one can use multiple side-channel parameters such as  $I_{DDT}$  and  $F_{max}$  together to make the Trojan effect more prominent [30]. The measured  $F_{max}$  values can be used to calibrate the interdie process corner of a chip. Any variation in  $I_{DDT}$ , which does not follow the expected trend due to process variations, may indicate presence of a Trojan. The detection sensitivity can be further improved through proper test vector selection or by using another parameter such as static current. One can also formulate it as a multimodal Trojan detection solution [41] to systematically consider the different parameters and increase the Trojan detection sensitivity.

5) *Test Generation for Trojan Detection*: Trojans are hard to detect using conventional testing mechanisms, since 100% scan-based fault coverage or full functional coverage cannot guarantee full manifestation of hidden Trojans. In software, methods that use a digital signature to validate the authenticity of the software before running it on the machine are popular [43]. However, at a chip level, there are no efficient notions of static signatures that can detect malicious intrusions. Hence, there is a need for test pattern generation techniques that can effectively detect Trojans of various forms and sizes. Next, we describe two vector generation techniques that exploit the properties of Trojans to detect them with high confidence.

The sustained vector technique [44] tries to drastically reduce the original circuit activity so that any incremental

activity in the triggered portion of the Trojan can be highlighted. However, one also needs to ensure that there is at least some kind of activity going on inside the circuit. In other words, the circuit should not be allowed to enter some sleep mode, which can make the Trojan dormant. On the other hand, in the statistical test generation approach, the circuit is first partitioned into smaller subcircuits, which are called regions [45]. The radius defines the extent of a region. Fig. 9 illustrates the concept of region and radius in a circuit. For a gate, the region around it comprises all the transitive fan-in and fan-out gates that are within the defined radius. Thus, a single gate constitutes a region of radius zero, immediate fan-in and fan-out gates along with the original gate constitutes a region of radius one, and so on. The regions are restricted across clock boundaries, i.e., no gates crossing flip-flops are included in a region. Once the regions have been defined, the technique attempts to create a minimal test set that maximizes the activity on a per-region basis.

6) *The Need for a Golden Model*: Side-channel-analysis-based Trojan detection typically depends on the existence of golden models or golden chips. Simulation-based analysis is typically not considered effective for process calibration due to potential inaccuracies in the simulation model and inherent uncertainties in the manufacturing process. The set of golden chips can be obtained in two ways: 1) by destructive reverse engineering to ensure that they are trusted; and 2) through exhaustive testing to verify their trustworthiness. Both processes are highly expensive and time consuming. Furthermore, it may not be practically feasible to comprehensively verify trustworthiness of even few chips through these processes. This has motivated researchers to develop side-channel-analysis-based countermeasures that do not require golden models. One such approach, called temporal self-referencing (TeSR), eliminates the requirement of golden chip instances by comparing a chip's transient current signature with itself, but at a different time window [72]. When a Trojan-free

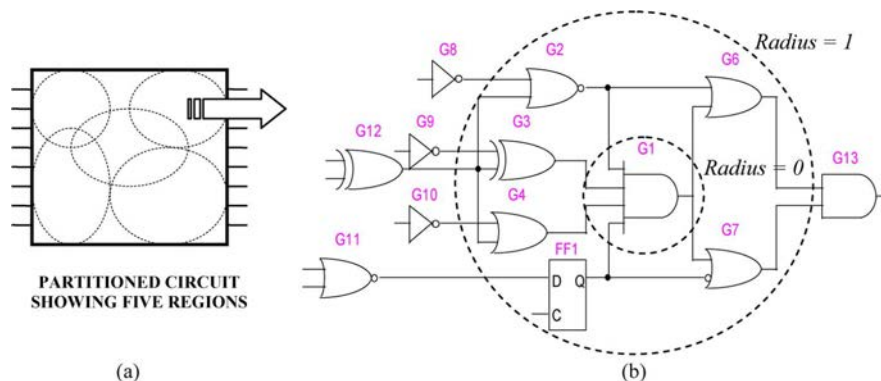


**Fig. 9.** Illustration of the concept of: (a) “region,” and (b) “radius” in a circuit during test vector generation [45].

circuit is made to undergo the same set of state transitions multiple times, the transient current “signature” should remain constant over different time windows. However, in a Trojan-infected circuit, the current signature varies over multiple time windows due to uncorrelated switching activity in the Trojan circuit. Such a temporal self-referencing approach can provide high detection sensitivity for Trojans of varying sizes. TeSR, however, applies only to sequential Trojans, and its effectiveness may be limited by the measurement noise. Structural self-similarity in a design can also be exploited to accomplish golden-free Trojan detection. The basic idea is to compare side-channel signature (e.g., current, delay) from a block (e.g., a full adder) with that from another similar block on the same chip. Natural symmetry in path delays in a circuit block can be used to efficiently isolate a Trojan-infected path from another by comparing delay signature of similar paths [73]. These self-referencing approaches rely on the fact that the Trojan effect is local and not all similar components are affected by a Trojan. In addition to not requiring a golden model, they automatically eliminate the effect of interdie process noise in a side-channel parameter, thus improving detection sensitivity [72].

### C. Trojan Detection in Hardware IPs

Today’s SoC designs use a large number of IP cores from different IP vendors, with varying degree of reliability associated with each vendor. Trustworthiness of these third-party IPs is imperative to ensure trustworthiness of the SoCs, in which they are used. Trojans may be inserted into IPs of different forms by a rogue designer or an untrusted CAD tool in an IP design house. Existing solutions for trusted IP acquisition fall into three broad classes. The first one relies on trust verification of IPs through directed test and verification. Conventional wisdom dictates modeling the IP trust verification problem as a formal verification problem. However, lack of a reliable golden design makes direct application of such verification approaches infeasible. One can, however, employ suspect-signal-guided sequential equivalence checking (SEC) [47] to gain some degree of confidence on the netlist. In the first step, functional vectors are generated using sequential ATPG to drop unsuspecting signals and identify suspect candidates. In the next step,  $N$ -detect full-scan ATPG is used to further discard the signals, which correspond to combinational untestable stuck-at-faults, by allowing the system to reach even those states, which are not functionally reachable. Then, the suspect circuit and specification-derived circuit are compared using FSM unrolling to see if the same behavior is produced when the suspect signal is activated. The approach, however, heavily relies on proper selection of the suspect signals. In [46], a structural checking approach is suggested to verify integrity of IPs, but the technique is not easily scalable to large designs. Recent works have also proposed using multiple copies of the same IP from different vendors to compare using unrolling [65] and validation [66] techniques.

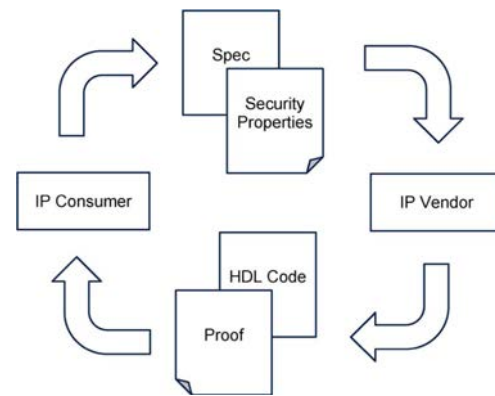


Fig. 10. IP acquisition and delivery protocol based on PCC [17].

They can identify malicious inclusions assuming that different vendors will not introduce identical Trojan instances.

The second class of approaches relies on a design paradigm for facilitating the acquisition of trustworthy IP. One such framework is based on proof-carrying code (PCC) [17]. Fig. 10 shows the protocol of trusted IP transfer between an IP writer and user. A set of security-related properties is formulated, and a formal proof of these properties is crafted by the designer. Any undesired modification to the IP is likely to violate the proofs. Validation of security-related properties is carried out by the IP user using the PCC to ensure that no hardware description language (HDL) code was modified or tampered with. Although the security-related properties cannot ensure complete trust in an IP vendor who crafted the formal proof of these properties, it nevertheless offers a line of defense against malicious alteration.

A new paradigm of expert-system-based IP analysis approaches [70], [71] forms the third class of solutions for IP trust verification. These approaches capture prior and trusted experience about designs, which include their specifications, into one or more knowledge bases (KBs). Next, KB rules are used to perform static/dynamic analysis to check correspondence to specifications. Primary challenge in these approaches lies in building the KBs. The knowledge for a standard IP can be captured by building specification ontologies, property-based models, and rule-base and associated assertion libraries. The manual effort is initially high and minimal afterwards. However, the reuse of KB across multiple IC design projects justifies the initial effort.

## V. DESIGN FOR SECURITY

Side-channel analysis and test generation for Trojan activation are widely discussed methods to detect Trojans.

These methods, though promising, must deal with major challenges due to rare activating nets in the circuit, process variations, and measurement noise. To improve the effectiveness of these detection methods, ICs must be designed with these detection strategies in mind. In fact, trust must be considered as an important design criterion in the design flow of modern ICs instead of being an afterthought.

#### A. Design Approaches to Prevent Trojan Insertion

Preventive approaches can be broadly classified into two major categories: 1) obfuscation-based approaches; and 2) layout-filler approaches. The first class of approaches is based on obscuring functional and structural properties of a design, thereby making it difficult for an attacker to incorporate Trojans [48], [49]. The central idea is to employ a key-based obfuscation technique that modifies state transition function of a given circuit. It makes a circuit operate in two distinct modes—the normal mode and the obfuscated mode [49]. While normal mode of operation results in desired output, obfuscated mode produces incorrect functional behavior for some input patterns. Such a modification obfuscates the rareness of the internal circuit nodes, thus making it difficult for an adversary to insert hard-to-detect Trojans. Fig. 11 illustrates the modification in the state transition diagram. It can also make some inserted Trojans benign since they can be active only in the obfuscated mode. The combined effect leads to higher Trojan detectability and higher level of protection against such attack. The approach, however, incurs hardware overhead and cannot provide high coverage against random Trojan insertion. Additionally, it requires application of a key to enable normal operation, which imposes the requirement for on-chip storage of key or disabling of the key by the designer after manufacturing.

The second class of approaches aims at denying opportunity for insertion of additional circuit components in a design by filling vacant spaces. To prevent an attacker

from identifying filled spaces in a layout and replace them with Trojan circuit, it is important to hide them appropriately. Built-in-self-repair (BISA) is a technique to fill all the unused spaces in an IC with functional standard cells instead of filler cells. Such an approach makes it challenging for an adversary to find any real estate on the device to insert Trojans [50]. BISA is a self-authenticating system, i.e., any attempt for tampering or changing the BISA architecture would be easy to detect. The flow involves identifying the vacant spaces on the design layout, placing the BISA cells to fill in the gaps, routing, and connecting these cells in such a way that they are self-authenticating after the fabrication. BISA provides protection against removal attacks where one or more of the filler standard cells are removed to place the Trojan cells by producing an incorrect signature during self-authentication. It also protects against redesign attacks and resizing attacks, using the same signature difference. Filler approaches including BISA, however, cannot prevent malicious alteration of a set of transistors or addition of a circuit that do not require extra layout space. Moreover, a resourceful adversary can redesign specific part of a circuit (e.g., resizing some transistors or constrained logic synthesis) to create room for Trojan insertion.

An emerging approach to prevent Trojan attack relies on a split-manufacturing process [51]. In this case, the front-end-of-the-line (FEOL) and back-end-of-the-line (BEOL) process steps are performed in separate fabrication facilities to hide the design intent and prevent malicious insertion. It considers fabrication of device layers (i.e., the FEOL steps) using an advanced process in an untrusted fabrication facility and high-level interconnects (i.e., the BEOL steps) to be processed using a less advanced trusted fabrication facilities. By not sharing the interconnection information to FEOL facilities, it prevents an attacker from understanding a design as well as from incorporating any undesired alteration since it requires connecting a set of transistors, e.g., realizing a Trojan circuit in Fig. 3(a), through a BEOL process.

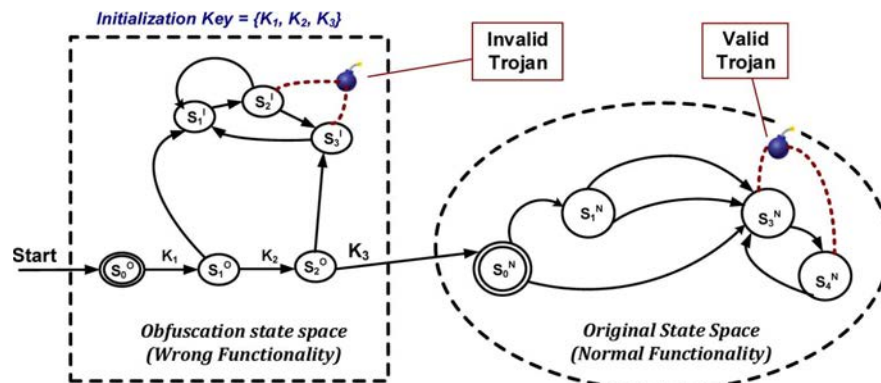


Fig. 11. Hardware-obfuscation-based design solutions hide design information to thwart Trojan attacks [49].

## B. Design Approaches to Facilitate Trojan Detection

Similar to testing for faults and functional bugs, Trojan detection can also benefit from specially crafted on-chip embedded structures. It needs to address the challenges due to rare activating nets in the circuit, process variations, and measurement noise [52]–[58]. Some of the major DFS approaches that aim at addressing them are described next.

1) *On-Chip Security Monitors*: The major challenge with side-channel analysis is that a small Trojan (e.g., of size 1–100 transistors) in a large multimillion transistor design would induce a barely noticeable effect in delay or supply current, even with the best available measuring instrument and judicious vector generation. In other words, the “detection sensitivity,” which is measured as the percentage deviation in delay or current due to a Trojan, is too low to identify a Trojan reliably. The problem of reduced detection sensitivity is greatly accentuated by the noise due to process and environmental fluctuations [52]. Researchers have proposed configuring circuit paths into ring oscillators [53], [54] to precisely detect small delay variations due to a Trojan. Moreover, integration of transient current sensors in a chip is shown to provide significantly higher detection sensitivity than conventional off-chip current monitoring [52]. Moreover, they provide a scalable solution for designs of arbitrary size and complexity as well as Trojans of various forms and sizes. One can exploit existing power-gating circuits used in modern designs to reduce the overhead of current sensors. Fig. 12(a) illustrates insertion of transient current monitors (CMs) in a chip. The detection sensitivity can improve significantly under process variations, as shown in Fig. 12(b). These on-chip structures, however, are themselves vulnerable to tampering by an adversary in an untrusted fabrication facility. Hence, they require

additional validation step to ensure their integrity. In [63], Waksman and Sethumadhavan have proposed two area-efficient detection mechanisms, called TRUSTNET and DATAWATCH, to detect attacks on microprocessors by implanting intelligent insiders within the chip.

2) *Removing Rare-Triggered Nets*: The stealthy nature of Trojans suggests that they connect to nets with low controllability and/or observability. As an example, a Trojan can have  $q > 1$  trigger inputs which can be nets with 1) very low transition probabilities; and 2) rare combinations. When the transition probability of  $\text{Net}_i$  is very low, either  $P_i(0) \gg P_i(1)$  or  $P_i(1) \gg P_i(0)$ . With  $q$  number of trigger inputs, the probability of generating a specific trigger vector is  $P_{\text{trigger-vector}} = \prod P_i$  ( $i = 1$  to  $q$ ), assuming independence of the nets. It is expected that  $P_{\text{trigger-vector}}$  is very low if  $P_i$ s are low. By increasing the transition probability of nets with low transition rate, it is possible to eliminate hard-to-activate sites in a design. Scan architecture allows access to internal cells of the circuits, thereby improving controllability and observability for internal nodes. To remove hard-to-activate sites, dummy scan flip-flops can be inserted to increase transition probability of design nets with transition probability less than a threshold ( $P_{th}$ ) [56]. Such design modifications can be done even if the gate-level netlist is not trusted.

3) *Increasing TCR*: While efficient vector generation can aim at achieving the goal of increasing TCR, as described earlier, one can achieve the same effect with appropriate design modification. It is possible to reorder scan cells based on their final physical location in the layout. Layout-aware scan-cell reordering can localize switching activity to one region while limiting it in other regions in a design

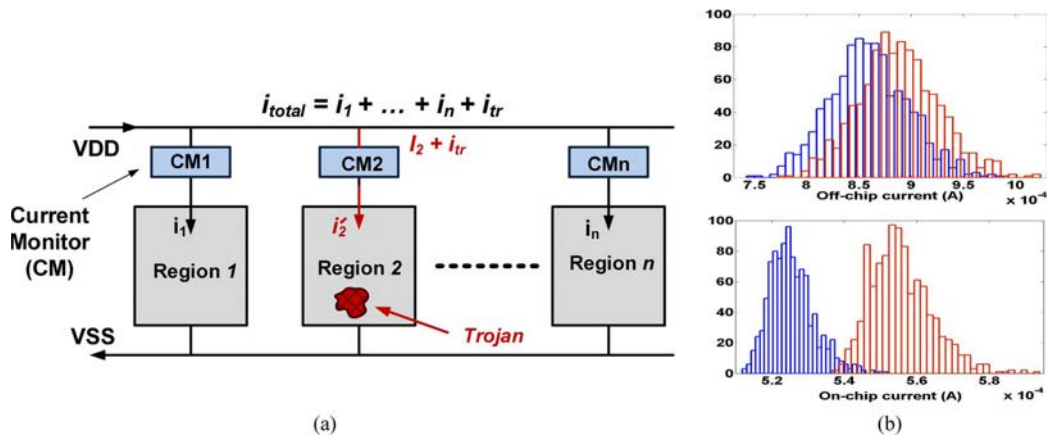


**Fig. 12.** (a) On-chip current monitors attached to power supply bumps can significantly improve the detection sensitivity in current-based side-channel analysis. (b) Comparison of off-chip versus on-chip current without (blue) and with Trojan (red) considering effect of process variations [52].

[55]. Simulation results show that this technique is very effective for small and large Trojans as well as distributed and localized Trojans. This is because reduction in circuit switching using this technique is substantially larger than the reduction in Trojan's switching, even if an attacker distributes the Trojan gates among different regions in the circuit. Contribution of Trojan to the total supply current can also be enhanced through voltage inversion [58]. Let us consider a four-input Trojan AND gate. When we invert the supply voltage of a CMOS gate, it behaves as its complement. Hence, once the voltage is inverted, the Trojan gate becomes a NAND gate and, for random input patterns, the triggering scenario occurs more frequently than in the normal (noninverted) operation. The approach is, however, effective for specific forms of Trojans, and suffers from scalability to large circuits.

## VI. RUNTIME MONITORING

Although detecting Trojans before ICs are deployed in the field is highly desirable, the existing techniques cannot guarantee comprehensive coverage of all types and sizes of Trojans. Hence, online monitoring of computations can significantly reduce the potential catastrophic effect of Trojan attacks. These approaches can be used to disable the chip upon detection of malicious logic or to bypass it and allow reliable operation, albeit with some performance overhead, in the presence of unreliable components. Next, we describe the major classes of these monitoring approaches.

### A. Configurable Security Monitors

One approach for such online monitoring is based on addition of reconfigurable logic in a given SoC to enable real-time functionality monitoring using security monitors (SMs) [6], [15]. Fig. 13 shows a SoC designed with SMs.

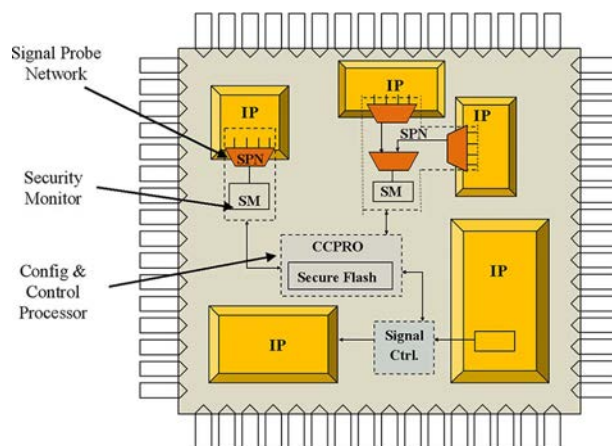


**Fig. 13. Runtime monitoring of Trojan effect using a reconfigurable infrastructure [20].**

SM is configured to implement finite state machines (FSMs) which check the behavior of signals of interest. These signals are fed to SMs through a signal probe network (SPN). The configuration of one SM does not interrupt the normal system operation or other SMs. The checks can be performed concurrently with the normal circuit operation and trigger appropriate countermeasures when a deviation from normal functionality is detected. SMs can be reconfigured by a configuration and control processor to dynamically implement different security checks to detect unexpected or illegal behavior created by a Trojan such as access to protected memory space or entering test mode during normal operation. Typically, the reconfigurable core will implement the functionality of the infected logic, which, in turn, will be disabled or bypassed.

### B. Variant-Based Parallel Execution

Another approach combines multicore hardware with dynamic distributed software scheduling to determine hardware trust during in-field use at runtime [31]. It involves scheduling and execution of functionally equivalent variants (obtained by different compilations, or different algorithm variations) simultaneously on different processing elements (PEs) capable of executing these variants and comparing the results. If a mismatch is detected, a new PE is engaged until a match is possible and the PEs with Trojans are identified. The idea is similar to N-version programming in software that generates and executes multiple functionally equivalent versions of the same program to achieve high reliability in presence of software faults. Even if two PEs can have similar Trojans, since they execute variants of a code, the Trojans are likely not to be activated simultaneously. It can achieve high level of trust for specific computer systems (e.g., many-core processor) at the cost of additional computation that can lead to performance and energy overhead. Effectiveness of such an approach also relies on efficient generation of variants.

### C. Hardware–Software Approach

In case of microprocessor-based systems, a software solution that can detect Trojan activation and/or tolerate Trojan effects can provide effective protection. For example, a simple verifiable “hardware guard” module external to the processor can be used for runtime execution monitoring to identify Trojan activation, as described in [23]. It targets primarily DoS and privilege escalation attacks using periodic checks by the OS, which is enhanced with live check functionality. It can be implemented with only 2.2% average performance overhead based on SPECint 2006 benchmark programs. A hybrid hardware/software approach referred to as *BlueChip* [24] is also investigated which includes a design-time component as well as runtime monitoring. It tries to identify any unused circuitry with design verification tests and tags it as suspicious. At runtime, the suspicious

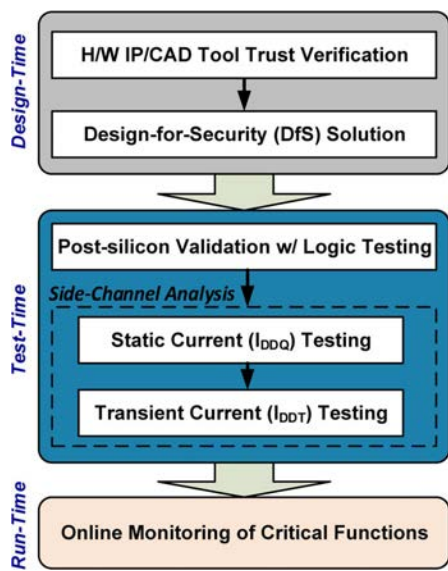


**Fig. 14.** Integrative protection approach that combines the benefits of design, test, and online monitoring solutions and hence can provide the highest level of confidence.

circuitry is removed and replaced with exception logic, which can trigger a software exception, allowing the system to perform normally by providing a detour around malicious hardware Trojans. This technique is designed to circumvent hardware Trojans, which are similar to software Trojans in purpose. These Trojans can be inserted in a hardware IP or in the instruction code running on an embedded processor.

## VII. SUMMARY AND FUTURE DIRECTIONS

The threat of hardware Trojan attacks during IC design and fabrication is escalating with increasing complexity of modern SoCs coupled with the evolving business model. Due to ever-growing computing demands, modern SoCs tend to include many heterogeneous processing cores (e.g., MPSoC), scalable communication network, together with reconfigurable cores, e.g., embedded field-programmable gate array (FPGA), in order to incorporate logic that is likely to change as standards and requirements evolve. Such design practices greatly increase the number of untrusted components in a SoC design. Growing reliance on untrusted third-party IPs, tools, and reduced control on the design/fabrication steps make modern ICs increasingly vulnerable to malicious manipulation. At the same time, new and more complex attack models, such as the ones that take advantage of collusion between IC design and test stages [21], are emerging. With growing awareness among the design as well as manufacturing communities on the

possibility of Trojan attacks, adversaries are likely to inflict unanticipated attacks, which are difficult to detect or prevent by existing design and test methods. Attacks such as leakage of confidential information through side channels [19] (e.g., power trace), selectively changing the process parameters, e.g., dopant polarity of transistors [60] during fabrication to compromise lifetime of an IC without affecting functional or parametric behavior, are some of the key examples. The latter does not require any additional logic element or interconnect. These attacks can be extremely stealthy and can bypass even very fine-grain detection techniques.

A “silver bullet” solution, which can reliably protect against Trojan attacks of all types, forms, and sizes is extremely difficult to achieve. On the other hand, an integrative solution, which combines the complementary benefits of design and test solutions, is expected to provide comprehensive coverage. Hence, protection approaches that integrate the benefits of a Trojan-aware design with efficient post-silicon test/validation as well as online monitoring can be attractive for high-assurance applications. Fig. 14 outlines the steps of such a cross-layer integrative protection approach. It includes trust verification of hardware IPs/tools, incorporating design-for-security measures in circuit blocks, postmanufacturing trust evaluation of ICs (through functional and parametric testing), and, finally, runtime monitoring of circuit operation. From the economic standpoint, however, it may be advantageous to invest more in the assurance of individual components (tools and IPs) than protecting the whole IC (through trust verification and monitoring) [59]. Emerging solutions such as nondestructive reverse engineering-based (e.g., through functional and side-channel analysis) trust validation, different forms of design obfuscation, and split-fabrication process [74], [75] are creating new pathways for protecting against these attacks.

The operating principles of emerging nanoscale devices can alter the concepts of hardware Trojan insertion and corresponding defense mechanisms. For example, the current flowing through nanoscale transistors strongly depends on channel stress, which can be modified through changes in process steps or even small layout changes, giving rise to new vulnerabilities. A more interesting problem will be to create reliability Trojans where malicious changes are engineered to radically accelerate device aging. Similar Trojans are also possible in noncharge-based devices, e.g., by marginally modulating spin polarization, or magnetic tunnel junction (MTJ) resistance. On the other hand, the level of intrinsic process variations in both charge- and noncharge-based emerging devices would affect the efficacy of side-channel analysis in detecting Trojan attacks.

Future work would focus on three major areas related to hardware Trojans: 1) exploring emerging attacks, in particular, attacks on circuits made with emerging

nanoscale devices; 2) developing a unified trust metric for coverage estimation, which can be extremely valuable in providing confidence level against arbitrary attacks; and 3) novel protection approaches. Future research on

protection approaches would include improving detection sensitivity for unknown Trojans and eliminating the need of golden ICs, Trojan-resilient design, and efficient online detection approaches. ■

## REFERENCES

- [1] S. Adee, "The hunt for the kill switch," *IEEE Spectrum*, vol. 45, no. 5, pp. 34–39, May 2008.
- [2] J. Kumagai, "Chip detectives," *IEEE Spectrum*, vol. 37, no. 11, pp. 43–48, Nov. 2000.
- [3] Defense Advanced Research Projects Agency (DARPA), "TRUST in integrated circuits (TIC)," 2007. [Online]. Available: <http://www.darpa.mil/MTO/solicitations/baa07-24>
- [4] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 296–310.
- [5] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 10–25, Jan.–Feb. 2010.
- [6] M. Abramovici and P. Bradley, "Integrated circuit security—New threats and solutions," in *Proc. 5th Annu. Workshop Cyber Security Inf. Intell. Res.*, 2009, DOI: 10.1145/1558607.1558671.
- [7] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware Trojan: Threats and emerging solutions," in *Proc. IEEE Int. High Level Design Validation Test Workshop*, 2009, pp. 166–171.
- [8] R. S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia, "MERO: A statistical approach for hardware Trojan detection," in *Proc. Workshop Cryptogr. Hardware Embedded Syst.*, 2009, pp. 396–410.
- [9] Chipworks Inc. "Semiconductor manufacturing—Reverse engineering of semiconductor components, parts and process," [Online]. Available: <http://www.chipworks.com>
- [10] Defense Science Board. "Task force on high performance microchip supply," 2005. [Online]. Available: <http://www.acq.osd.mil/dsb/reports/ADA435563.pdf>
- [11] Australian Government, Department of Defence (DoD), Defence Science and Technology Organisation (DSTO). "Towards countering the rise of the silicon Trojan," 2008. [Online]. Available: <http://dspace.dsto.defence.gov.au/dspace/bitstream/1947/9736/1/DSTO-TR-2220%20PR.pdf>
- [12] A. Rawnsley, "Fishy chips: Spies want to hack-proof circuits," *Wired*, Jun. 24, 2011. [Online]. Available: <http://www.wired.com/dangerroom/2011/06/chips-oy-spies-want-to-hack-proof-circuits/>
- [13] R. Johnson, "The Navy bought fake Chinese microchips that could have disarmed U.S. missiles," *Business Insider*, Jun. 27, 2011. [Online]. Available: <http://www.businessinsider.com/navy-chinese-microchips-weapons-could-have-been-shut-off-2011-6>
- [14] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," *Cryptographic Hardware and Embedded Systems Workshop*, vol. 7428, Berlin, Germany: Springer-Verlag, 2012, pp. 23–40.
- [15] S. Bhunia et al., "Protection against hardware Trojan attacks: Towards a comprehensive solution," *IEEE Design Test Comput.*, vol. 30, no. 3, pp. 6–17, May–Jun. 2013.
- [16] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi, "A taxonomy of computer program security flaws," *ACM Comput. Surv.*, vol. 26, no. 3, pp. 211–254, 1994.
- [17] E. Love, Y. Jin, and Y. Makris, "Proof-carrying hardware intellectual property: A pathway to trusted module acquisition," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 25–40, Feb. 2012.
- [18] M. Potkonjak, "Synthesis of trustable ICs using untrusted CAD tools," in *Proc. Design Autom. Conf.*, 2010, pp. 633–634.
- [19] L. Lin, W. Burleson, and C. Paar, "MOLES: Malicious off-chip leakage enabled by side-channels," in *Proc. Int. Conf. Comput.-Aided Design*, 2009, pp. 117–122.
- [20] M. Abramovici, "Protecting integrated circuits from silicon Trojan horses," *Military Embedded Syst.*, Jan.–Feb., 2009. Available: <http://www.mil-embedded.com/articles/id/73748>
- [21] S. Ali, D. Mukhopadhyay, R. S. Chakraborty, and S. Bhunia, "Multi-level attack: An emerging threat model for cryptographic hardware," in *Proc. Design Autom. Test Eur. Conf. Exhibit.*, 2011, DOI: 10.1109/DATE.2011.5763307.
- [22] S. Jha and S. K. Jha, "Randomization based probabilistic approach to detect Trojan circuits," in *Proc. IEEE High Assurance Syst. Eng. Symp.*, 2008, pp. 117–124.
- [23] G. Bloom, B. Narahari, and R. Simha, "OS support for detecting Trojan circuit attacks," in *Proc. IEEE Int. Workshop Hardware-Oriented Security Trust*, 2009, pp. 100–103.
- [24] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith, "Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically," in *Proc. IEEE Symp. Security Privacy*, 2010, pp. 159–172.
- [25] S. T. King et al., "Designing and implementing malicious hardware," in *Proc. 1st USENIX Workshop Large-Scale Exploits Emergent Threats*, 2008, article 5.
- [26] X. Wang, S. Narasimhan, A. Krishna, T. Mal-Sarkar, and S. Bhunia, "Software exploitable hardware Trojan attacks in embedded processor," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, 2012, pp. 55–58.
- [27] Y. Shiyanovskii et al., "Process reliability based trojans through NBTI and HCI effects," in *Proc. NASA/ESA Conf. Adaptive Hardware Syst.*, 2010, pp. 215–222.
- [28] F. Kiamilev and R. Hoover, "Demonstration of hardware Trojans," presented at the DEFCON 16, Las Vegas, NV, USA, Aug. 8–10, 2008.
- [29] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware Trojans," *IEEE Computer*, vol. 43, no. 10, pp. 39–46, Oct. 2010.
- [30] S. Narasimhan et al., "Hardware Trojan detection by multiple-parameter side-channel analysis," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2183–2195, Nov. 2013.
- [31] D. McIntyre, F. Wolff, C. Papachristou, and S. Bhunia, "Dynamic evaluation of hardware trust," in *Proc. IEEE Int. Workshop Hardware-Oriented Security Trust*, 2009, pp. 108–111.
- [32] J. A. Kash, J. C. Tsang, and D. R. Knebel, "Method and apparatus for reverse engineering integrated circuits by monitoring optical emission," U.S. Patent 6 496 022 B1, 2002.
- [33] M. Banga, M. Chandrasekar, L. Fang, and M. Hsiao, "Guided test generation for isolation and detection of embedded Trojans in ICs," in *Proc. 18th ACM Great Lakes Symp. VLSI*, 2008, pp. 363–366.
- [34] J. Aarestad, D. Acharyya, R. Rad, and J. Plusquellic, "Detecting Trojans through leakage current analysis using multiple supply pad IDDQs," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 4, pp. 893–904, Dec. 2010.
- [35] Y. Alkabani and F. Koushanfar, "Consistency-based characterization for IC Trojan detection," in *Proc. Int. Conf. Comput.-Aided Design*, 2009, pp. 123–127.
- [36] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, "Hardware Trojan horse detection using gate-level characterization," in *Proc. Design Autom. Conf.*, 2009, pp. 688–693.
- [37] D. Acharyya and J. Plusquellic, "Calibrating power supply signal measurements for process and probe card variations," in *Proc. IEEE Int. Workshop Current Defect Based Test.*, 2004, pp. 23–30.
- [38] D. Rai and J. Lach, "Performance of delay-based Trojan detection techniques under parameter variations," in *Proc. IEEE Int. Workshop Hardware-Oriented Security Trust*, 2009, pp. 58–65.
- [39] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *Proc. IEEE Int. Workshop Hardware-Oriented Security Trust*, 2008, pp. 51–57.
- [40] C. Lamech, R. M. Rad, M. Tehranipoor, and J. Plusquellic, "An experimental analysis of power and delay signal-to-noise requirements for detecting Trojans and methods for achieving the required detection sensitivities," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pt. 2, pp. 1170–1179, Sep. 2011.
- [41] F. Koushanfar and A. Mirhoseini, "A unified framework for multimodal submodular integrated circuits Trojan detection," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 1, pp. 162–174, Mar. 2011.
- [42] D. Du, S. Narasimhan, R. S. Chakraborty, and S. Bhunia, "Self-referencing: A scalable side-channel approach for hardware Trojan detection," in *Proc. 12th Int. Conf. Cryptogr. Hardware Embedded Syst. Workshop*, 2010, pp. 173–187.
- [43] M. A. Williams, "Anti-Trojan, Trojan detection with in-kernel digital signature testing of executables," *Security Software Engineering: NetXSecure NZ Limited, Tech. Rep.*, 2002, pp. 1–12.
- [44] M. Banga and M. Hsiao, "A novel sustained vector technique for the detection of hardware Trojans," in *Proc. 22nd Int. Conf. VLSI Design*, 2009, pp. 327–332.
- [45] M. Banga and M. Hsiao, "A region based approach for the detection of hardware Trojans," in *Proc. IEEE Symp. Hardware-Oriented Security Trust*, 2008, pp. 40–47.
- [46] S. Smith and J. Di, "Detecting malicious logic through structural checking," in *Proc. IEEE Region 5 Tech. Conf.*, 2007, pp. 217–222.
- [47] M. Banga and M. Hsiao, "Trusted RTL: Trojan detection methodology in pre-silicon designs," in *Proc. IEEE Int. Workshop Hardware-Oriented Security Trust*, 2010, pp. 56–59.

- [48] J. Rajendran et al., "Securing processors against insider attacks: A circuit-microarchitecture co-design approach," *IEEE Design Test*, vol. 30, no. 2, pp. 35–44, Mar.–Apr. 2013.
- [49] R. S. Chakraborty and S. Bhunia, "Security against hardware Trojan attacks using key-based design obfuscation," *J. Electron. Testing, Theory Appl.*, vol. 27, no. 6, pp. 767–785, Dec. 2011.
- [50] K. Xiao and M. Tehranipoor, "BISA: Built-in-self-authentication for preventing hardware Trojan insertion," in *Proc. IEEE Int. Workshop Hardware-Oriented Trust Security*, 2013, pp. 45–50.
- [51] F. Imeson, A. Emtenan, S. Garg, and M. V. Tripunitara, "Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation," in *Proc. 22nd USENIX Conf. Security*, 2013, pp. 495–510.
- [52] S. Narasimhan, W. Yueh, X. Wang, S. Mukhopadhyay, and S. Bhunia, "Improving IC security against Trojan attacks through integration of security monitors," *IEEE Design Test Comput.*, vol. 29, no. 5, pp. 37–46, Oct. 2012.
- [53] J. Rajendran, V. Jyothi, O. Sinanoglu, and R. Karri, "Design and analysis of ring oscillator based Design-for-trust technique," in *Proc. IEEE 29th VLSI Test Symp.*, 2011, pp. 105–110.
- [54] X. Zhang and M. Tehranipoor, "RON: An on-chip ring oscillator network for hardware Trojan detection," in *Proc. Design Test Eur. Conf. Exhibit.*, 2011, DOI: 10.1109/DATE.2011.5763260.
- [55] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A layout-aware approach for improving localized switching to detect hardware Trojans in integrated circuits," in *Proc. IEEE Int. Workshop Inf. Forensics Security*, 2010, DOI: 10.1109/WIFS.2010.5711438.
- [56] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A novel technique for improving hardware Trojan detection and reducing Trojan activation time," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 112–125, Jan. 2012.
- [57] M. Banga and M. Hsiao, "ODETTE: A non-scan design-for-test methodology for Trojan detection in ICs," in *Proc. IEEE Int. Workshop Hardware-Oriented Trust Security*, 2011, pp. 18–23.
- [58] M. Banga and M. Hsiao, "VITAMIN: Voltage inversion technique to ascertain malicious insertions in ICs," in *Proc. IEEE Int. Workshop Hardware-Oriented Trust Security*, 2009, pp. 104–107.
- [59] J. Grossklags, N. Christin, and J. Chuang, "Secure or insecure?: A game-theoretic analysis of information security games," in *Proc. 17th Int. Conf. World Wide Web*, 2008, pp. 209–218.
- [60] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Bursleson, "Stealthy dopant-level hardware Trojans: Extended version," *J. Cryptogr. Eng.*, vol. 4, no. 1, pp. 1–13, Apr. 2014.
- [61] S. Wei and M. Potkonjak, "Scalable hardware Trojan diagnosis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 6, pp. 1049–1057, Jun. 2012.
- [62] W. Zhang et al., "Virtual probe: A statistical framework for low-cost silicon characterization of nanoscale integrated circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 12, pp. 1814–1827, Dec. 2011.
- [63] A. Waksman and S. Sethumadhavan, "Tamper evident microprocessors," in *Proc. IEEE Symp. Security Privacy*, 2010, pp. 173–188.
- [64] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: Identification of stealthy malicious logic using Boolean functional analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2013, pp. 697–708.
- [65] T. Reece, D. B. Limbrick, and W. H. Robinson, "Design comparison to identify malicious hardware in external intellectual property," in *Proc. IEEE 10th Int. Conf. Trust Security Privacy Comput. Commun.*, Changsha, China, 2011, pp. 639–646.
- [66] C. Liu, J. Rajendran, C. Yang, and R. Karri, "Shielding heterogeneous MPSoCs from untrustworthy 3PIPs through security-driven task scheduling," in *Proc. IEEE Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, 2013, pp. 101–106.
- [67] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi, "A taxonomy of computer program security flaws," *ACM Comput. Surv.*, vol. 26, no. 3, pp. 211–254, 1994.
- [68] L. A. Hughes and G. J. DeLone, "Viruses, worms, and Trojan Horses: Serious crimes, nuisance, or both?" *Social Sci. Comput. Rev.*, vol. 25, no. 1, pp. 78–98, 2007.
- [69] G. McGraw and G. Morrisett, "Attacking malicious code: A report to the Infosec Research Council," *IEEE Software*, vol. 17, no. 5, pp. 33–41, Sep.–Oct. 2000.
- [70] B. Singh et al., "Knowledge-guided methodology for third-party soft IP analysis," in *Proc. 27th Int. Conf. VLSI Design*, 2014, pp. 246–251.
- [71] B. Singh et al., "Cross-correlation of specification and RTL for soft IP analysis," in *Proc. Design Autom. Test Eur. Conf. Exhibit*, 2014, DOI: 10.7873/DATE2014.303.
- [72] S. Narasimhan, X. Wang, D. Du, R. S. Chakraborty, and S. Bhunia, "TeSR: A robust temporal self-referencing approach for hardware Trojan detection," in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2011, pp. 71–74.
- [73] N. Yoshimizu, "Hardware Trojan detection by symmetry breaking in path delays," in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2014.
- [74] R. W. Jarvis and M. G. McIntyre, "Split manufacturing method for advanced semiconductor circuits," U.S. Patent 7 195 931, 2004.
- [75] J. V. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Proc. Design Autom. Test Eur. Conf. Exhibit.*, 2013, pp. 1259–1264.
- [76] TrojanHunter help file. [Online]. Available: <http://www.trojanhunter.com/trojanhunter/help/>
- [77] Y. Liu, Y. Jin, and Y. Makris, "Hardware Trojans in wireless cryptographic ICs: Silicon demonstration & detection method evaluation," in *Proc. Int. Conf. Comput.-Aided Design*, 2013, pp. 399–404.

## ABOUT THE AUTHORS

**Swarup Bhunia** (Senior Member, IEEE) received the B.E. degree (honors) from Jadavpur University, Kolkata, India, in 1995, the M.Tech. degree from the Indian Institute of Technology (IIT), Kharagpur, India, in 1998, and the Ph.D. degree from Purdue University, West Lafayette, IN, USA, in 2005.

Currently, he is a T.A. Schroeder Associate Professor of Electrical Engineering and Computer Science at Case Western Reserve University, Cleveland, OH, USA. He has over ten years of research and development experience with over 150 publications in peer-reviewed journals and premier conferences in the area of very large scale integration (VLSI) design, computer-aided design (CAD), and test techniques. His research interests include low-power and robust design, hardware security and protection, adaptive nanocomputing and novel test methodologies. He has worked in the semiconductor industry on register transfer level (RTL) synthesis, verification, and low-power design for about three years.

Dr. Bhunia received the IBM Faculty Award (2013), the National Science Foundation (NSF) CAREER development award (2011), the Semiconductor Research Corporation (SRC) Inventor Recognition Award (2009), the SRC technical excellence award (2005), and several best paper awards/nominations. He has served as a Guest Editor of IEEE



DESIGN & TEST OF COMPUTERS (2010, 2012), Guest Editor of the IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS (2014), and on the Editorial Board of the *Journal of Low Power Electronics*. He has served as Program Chair for the 2013 IEEE/ACM NANOARCH and the 2014 International Symposium on VLSI Design and Test (VDAT), and will serve as Program Chair for the 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). He served in the technical program committees of major IEEE and Association for Computing Machinery (ACM) conferences.

**Michael S. Hsiao** (Fellow, IEEE) received the B.S. degree in computer engineering (highest honors) and the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 1992, 1993, and 1997, respectively.

Currently, he is a Professor in the Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA. His current research interests include design, testing, verification, and trust of hardware and software.



Prof. Hsiao was a recipient of the Digital Equipment Corporation Fellowship, the McDonnell Douglas Scholarship, and the National Science Foundation CAREER Award. Among his publications, he has been recognized for the most influential papers in the first ten years (1998–2007) of Design Automation and Test Conference in Europe (DATE). He received the best paper award at the 2010 IEEE Asian Test Symposium and the best student paper award at the 2012 IEEE International Test Conference.

Prof. Hsiao was an Associate Editor for the IEEE TRANSACTIONS ON COMPUTERS, the *ACM Transactions on Design Automation of Electronic Systems*, and IEEE DESIGN & TEST OF COMPUTERS. He served as Program Chair for the 2014 IEEE Hardware Oriented Security and Trust Symposium, in addition to serving on technical program committees for numerous conferences.

**Mainak Banga** (Member, IEEE) received the B.E. degree in electronics and communication engineering (Gold Medalist) from Birla Institute of Technology, Mesra, India, in 2003 and the M.S. and Ph.D. degrees in computer engineering from the Virginia Polytechnic Institute and State University, Blacksburg, VA, in 2008 and 2010, respectively.

Currently, he is a Software Engineer with Intel Corporation, Folsom, CA, USA, in the System Validation and Engineering Group. He is involved in developing software tools to aid pre- and post-silicon debug. He worked extensively on Trojan detection mechanisms in pre- and post-silicon IC design flows in his graduate research under the guidance of Prof. M. S. Hsiao. His publications have been selected in renowned conferences like Design Automation and Testing in Europe (DATE), Hardware-Oriented Security and Trust (HOST), VLSI Design, Asian Test



Symposium, and Great Lake Symposium on VLSI. His research interest also includes creating faster and more efficient methodologies for design for test.

**Seetharam Narasimhan** received the B.E. (honors) degree in electronics and telecommunication engineering from Jadavpur University, Kolkata, India, in 2006 and the Ph.D. degree in computer engineering from Case Western Reserve University, Cleveland, Ohio, USA, in 2012.

Currently, he is a Component Design Engineer with Intel Corporation, Hillsboro, OR, USA, in the System Validation and Engineering (SVE) Security Center of Excellence. His areas of interest include hardware security evaluation of system-on-chip products. He has worked extensively on hardware Trojan design and detection as part of his graduate research under the guidance of Prof. S. Bhunia. He has two book chapters and more than 30 publications in peer-reviewed journals and premier conferences in the areas of hardware security and biomedical very large scale integration (VLSI) design.

Dr. Narasimhan has served as a peer reviewer for various IEEE conferences and journals and in the Technical Program Committee for several IEEE/ACM conferences such as the Design Automation Conference (DAC), the International Symposium on Quality Electronic Design (ISQED), VLSI Design-India, the Workshop on NanoArchitecture (NANOARCH), and the International Conference on Computer Design (ICCD). He has been a student competition finalist at the IEEE EMBS conference in 2009, finalist at the CSAW Embedded Systems Challenge in 2009–2011, and received the best paper nomination at the 2010 IEEE Symposium on Hardware-Oriented Security and Trust (HOST).



# FPGA Security: Motivations, Features, and Applications

*This paper discusses all aspects of FPGA security and trust.*

By STEPHEN M. TRIMBERGER, *Fellow IEEE*, AND JASON J. MOORE

**ABSTRACT** | Since their inception, field-programmable gate arrays (FPGAs) have grown in capacity and complexity so that now FPGAs include millions of gates of logic, megabytes of memory, high-speed transceivers, analog interfaces, and whole multicore processors. Applications running in the FPGA include communications infrastructure, digital cinema, sensitive database access, critical industrial control, and high-performance signal processing. As the value of the applications and the data they handle have grown, so has the need to protect those applications and data. Motivated by specific threats, this paper describes FPGA security primitives from multiple FPGA vendors and gives examples of those primitives in use in applications.

**KEYWORDS** | Anti-tamper (AT); authentication; encryption; field-programmable gate arrays (FPGAs); information assurance; physically uncloneable function (PUF); system on chip (SoC); trust

## I. INTRODUCTION

### A. FPGAs and Programming Technology

A field-programmable gate array (FPGA) is a semiconductor device that can be programmed after manufacture to perform a specific application design, typically specified as a digital logic system [43]. A taxonomy of FPGAs commonly starts with the program storage technology (Fig. 1).

SRAM-programmed FPGAs store their configuration data in internal volatile memory cells distributed throughout the device. These are generally not SRAM cells, but are more similar to static latch cells [43]. Xilinx's 7-Series and Altera's Stratix-5 are examples of popular SRAM-based FPGAs. A recognized disadvantage of SRAM programming

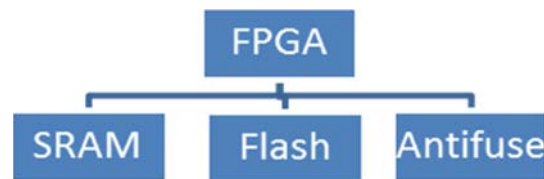


**Fig. 1.** FPGA taxonomy.

stems from its volatility. When power is removed, the programming is lost, so an SRAM FPGA requires an external nonvolatile memory for permanent storage of the application program. When power is applied, the SRAM FPGA loads its programming bitstream from that external storage. Besides requiring a second device, the transmission of the program from the nonvolatile external memory to the SRAM FPGA may expose the programming to a potential adversary. The volatility of data may also be used as a positive security feature, enabling the SRAM FPGA to clear all programming if it is tampered [48].

In contrast, flash memory programmable logic devices, such as traditional complex programmable logic devices (CPLDs), the Microsemi Corporation (Aliso Viejo, CA, USA) SmartFusion2, and Lattice Semiconductor (Hillsboro, OR, USA) ispXPGA [1], [21], are nonvolatile and use internal flash memory to hold the programming. While the internal flash memory eliminates the need for an external nonvolatile storage device and the consequent exposure of the programming to potential adversaries, systems employing flash FPGAs commonly require in-system programming (ISP) of the FPGA. ISP exposes the programming of the FPGA to the same security concerns as SRAM FPGAs. The availability of reprogrammable flash provides FPGA manufacturers with the ability to build applications that “remember” information through power cycles—useful in cryptographic applications such as tamper logging and key revocation. Flash devices can also be erased upon command to eliminate the design when needed.

Antifuse FPGAs, such as the Microsemi Axcelerator, use a one-time programmable structure to form

Manuscript received September 14, 2013; revised May 16, 2014; accepted June 11, 2014. Date of publication July 8, 2014; date of current version July 18, 2014.

**S. M. Trimberger** is with Xilinx, San Jose, CA 95124 USA (e-mail: steve.trimberger@xilinx.com).

**J. J. Moore** is with Xilinx, Albuquerque, NM 87109 USA.

Digital Object Identifier: 10.1109/JPROC.2014.2331672

0018-9219 © 2014 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See [http://www.ieee.org/publications\\_standards/publications/rights/index.html](http://www.ieee.org/publications_standards/publications/rights/index.html) for more information.

nonvolatile links between internal nodes [2], [25], [26]. An antifuse, commonly built as a programmable via between metal layers, is disconnected at manufacture. A high-voltage pulse programs the fuse, causing it to form a low-resistance connection between the internal nodes. Antifuses are nonvolatile, but one-time programmable. Once programmed, an antifuse FPGA cannot be changed or reprogrammed. Because there is no need for external configuration storage, the confidentiality and authentication of configuration data is more easily maintained. ISP is not possible. However, because the program cannot be erased from the device, additional system-level security concerns may remain.

### B. Why SRAM?

By far, the most common FPGAs, even in security-conscious applications, are those programmed with SRAM. If SRAM programming exposes sensitive data to adversaries, why would anyone use them? The popularity of SRAM programming technology derives from the simplicity of its manufacture: SRAM FPGAs require only transistors and wires to realize the interconnect, configuration memory cells and switches of the generic device. Therefore, SRAM FPGAs take advantage of new process nodes earlier than other FPGAs [47], which may be two process generations ahead of other technologies. This process advantage results in higher performance, greater logic density, and improved power efficiency for SRAM FPGAs. SRAM programming also simplifies manufacturing test, where the SRAM FPGA is typically programmed many times to perform self-tests. In addition, SRAM FPGA applications can be easily updated in the field in much the same way software is updated.

When used with strong bitstream security features, including those described in this paper, the security of SRAM FPGAs is on par with the security of nonvolatile internal storage of the bitstream. Therefore, despite the greater perceived security of antifuse and flash FPGAs, SRAM FPGAs are deployed in many security-conscious applications.

### C. The FPGA Design Lifecycle

The FPGA lifecycle includes two design flows: the base array design and the application design (Fig. 2). Security must be maintained through both [44]. The base array design is a standard integrated circuit development flow controlled by the FPGA manufacturer. The base array is designed using commercial design tools and libraries, manufactured at a foundry and tested. It is then typically sent to another facility for packaging and final test. The resulting base array is shipped to a customer or authorized distributor. The base array design is subject to all the supply chain trust and security concerns as any other integrated circuit, including questions about tampering with tools, supply-chain control, and reverse engineering. Large FPGA manufacturers maintain a close watch on their

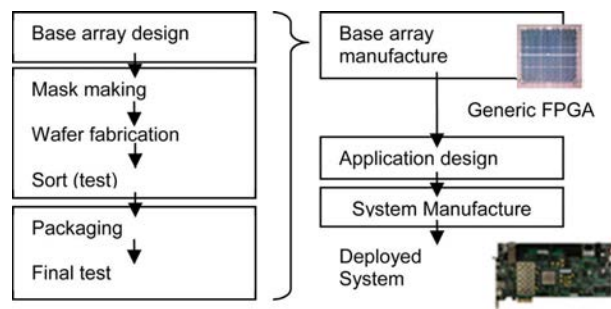


**Fig. 2. FPGA lifecycle flows. (Left) Generic integrated circuit flow for the base array. (Right) Application design and deployment flow.**

supply chain, tracking every manufactured device through to final customer delivery or destruction. As the security issues associated with the design and manufacture of the base array are no different than those of other semiconductor devices, this paper does not focus on the base array design and manufacture, but instead focuses on the security concerns that arise from the need to protect the application design.

The application design also has a design phase, typically performed with FPGA vendors' tools, often augmented with commercial EDA tools. The application developer integrates design information or intellectual property (IP) from a number of sources into an FPGA application: original and reused hardware description language (HDL) code, libraries from the FPGA vendor and other parties and software for soft and hard microprocessors. The FPGA vendor's tools compile the application design into a bitstream, the programming of the FPGA base array to realize the application function. As with any design process, the design itself can be carried out in a secure location. Protection of IP during the design phase is no different for FPGAs than it is for ASICs or microprocessors. Therefore, this paper does not address design-phase security.

A nonvolatile FPGA, such as a flash or antifuse FPGA, may be programmed before it is shipped. An SRAM FPGA is typically shipped with a separate nonvolatile memory containing the programming, and when power is applied, the FPGA loads its programming from the nonvolatile memory.

### D. This Paper

This paper begins by focusing on those FPGA aspects that impact security, both positively and negatively. It summarizes the common threat vectors and then introduces some early FPGA security strategies. The remainder of the paper focuses on modern FPGA security as it relates to two of the primary security domains: information assurance (IA) and anti-tamper (AT). In each domain, the presentation describes the techniques that are currently deployed, introducing them broadly, then using specific threats to motivate additional detail. The various FPGA

vendors have chosen solutions to these threats that are similar, yet they differ in detail. In this paper, we attempt to describe the major security solutions deployed by large FPGA vendors, outlining major distinctions while omitting minor differences. Since FPGA security is continually changing, newer FPGAs may well deploy different mechanisms.

Following the discussion of security features is a section showing applications using those features to achieve security goals. This paper concludes with a short discussion of the future of FPGA security capabilities.

## II. FPGA SECURITY INTRODUCTION

### A. Unique Aspects of FPGA Security

FPGA programming bitstreams are qualitatively much like microprocessor software. They are susceptible to all the same security concerns that surround software, including unauthorized copy, theft of IP embodied in the FPGA application program, and tampering to introduce malware [9], [46]. FPGA programming is present in the system in the field, whether programmed directly in antifuses, flash memory cells, or in an external nonvolatile memory. If an adversary can recover the programming by reading the internal memory, intercepting the programming bitstream, or reverse-engineering programmed fuses from a decapped device, then the application can be duplicated and reverse engineered. SRAM FPGAs, in particular, have been criticized over this concern [2], although Flash-based FPGAs have the same susceptibility if in system reprogrammability is required.

On the other hand, the application developer does not reveal the application design to FPGA vendors or their suppliers. Because the FPGA base array is manufactured without knowledge of the end application, there is no chance of IP theft or tampering of an application design during manufacture and test of the FPGA base array. Since all FPGA devices are manufactured identically and sold into a variety of applications, an adversary cannot discover any application-dependent information by attacking the FPGA vendor's supply chain.

Further, since the programming is not done with metallization as is the case with ASIC devices, traditional reverse engineering, where the mask layers are recognized from a decapped device, does not work. Such reverse engineering may yield the application-independent base array, but not the application implemented on it.

### B. Environment and the Cost of Security

FPGA security is complicated by the environment in which the FPGA is expected to perform. The design of FPGA security features assumes no physical barrier and no communication network: the FPGA may be in the hands of an adversary with no trusted party available. This environmental assumption distinguishes FPGA security from

internet security, where servers may physically reside in a trusted environment and those servers can verify identity through name servers with which they are in communication. In FPGA security design, it is assumed that the adversary has physical access to the device and may mount any electrical, physical, side channel, or replay attack. The rationale is straightforward: if the adversary does not have such access, then the containing system could ensure the security of the FPGA by controlling all access to the FPGA. In this case, built-in FPGA security would be redundant.

Although military systems may employ physical security, the cost of "guns, gates, and guards" is impractical in commercial systems. The adversary is assumed to have an economic motive, such as theft of IP. Therefore, the security applied in the commercial domain is an economic concern where the cost of security measures is balanced against the value of the information being protected. FPGA security is designed to make the cost of breaking the security greater than the adversary's expected economic gain. This decision is ultimately in the hands of the application developer, not the FPGA manufacturer.

As FPGAs have become larger and more capable, the value of the IP of the application designs has grown, motivating significant investment in built-in security functions. Further, the value of the data handled by the FPGA has also increased significantly, including such information as decrypted digital cinema and personal-data databases. As a result, today we find FPGAs deployed in a security-hostile environment, protecting data of great commercial value.

## III. THREATS

An adversary may attack the IP of the application design itself, the data stored in the application or the system of which the FPGA is a part. Each type of data has different value. Each attack requires different security features to defend. The attacks of major concern to FPGA vendors can be divided into categories.

### A. Cloning/Overbuilding

In cloning, an adversary copies the FPGA programming, then uses it in an identical device, selling it as his own. In overbuilding, an adversary such as a contract manufacturer builds additional systems, inserting the legitimate bitstream into those systems and selling them without the designer's approval. Cloning may apply to an entire design or may apply to a subset of the design, for example, purchased cores that may be restricted by the seller. In both cases, the adversary does not require detailed knowledge of the design.

### B. Reverse Engineering

An adversary may reverse engineer the bitstream to recover the circuit design that it implements. This may be done to understand and duplicate the functionality of that

application, but may also be used as part of an attack on other aspects of the system. Reverse engineering may be used to tamper with the application to insert malware. Historically, reverse engineering an FPGA bitstream, like decompiling software, has been considered possible, though tedious and nontrivial. Reverse engineering of FPGA bitstreams is further complicated because FPGA vendors do not have a standardized bitstream. As a result, every new FPGA device requires a new bitstream reverse-engineering effort.

A more insidious problem is dealing with the size of the application. Although reverse engineering may divulge the netlist of the application, transforming a multimillion gate netlist into an understandable design that can be modified is problematic. The complexity of the application increases its value, making theft attractive, but the consequent size makes theft difficult.

Regardless, researchers have periodically reported the ability to reverse engineer unencrypted bitstreams. It would seem imprudent to rely on the tedium of bitstream reverse engineering to protect valuable IP.

### C. Tampering

In tampering, an adversary modifies an application design. Tampering may be employed to add logic that leaks information from an application or tampering may disable parts of the application, potentially defeating other security measures. For the former, tampering must control the application to set values in the bitstream, so reverse engineering may also be required. However, for the latter, merely scrambling parts of the bitstream may be sufficient.

### D. Spoofing

In spoofing, an adversary replaces the FPGA bitstream with his own. That bitstream may or may not include components derived from cloning or reverse engineering. A spoofed application may compromise the system in which it operates.

### E. Denial of Service, Destruction of the FPGA, and Substitution

Since it is assumed that the FPGA is in the hands of an adversary, denial of service and malicious destruction of the FPGA device are somewhat irrelevant. Rather than mount a clever attack on the design to prevent the system from operating, an adversary could simply smash the FPGA with a hammer. Conversely, if a system requires an FPGA containing a unique key, an adversary may choose to circumvent security measures by replacing the FPGA in a system with another identically manufactured device from the FPGA vendor without the key or with his own key. In many cases, this substitution is simpler than attempting to break the FPGA device security. Since these physical attacks are so simple, FPGAs typically do not defend against these types of threats.

## IV. HISTORICAL FPGA SECURITY

Early FPGAs contained very little logic, and by inference that logic had low value. Therefore, when they were introduced, FPGAs provided only rudimentary protection against threats.

FPGA manufacturers did not release the coding of their bitstreams, though they did release a considerable amount of information about the bitstream in tools and documentation [50]. They considered the task of reverse engineering the bitstream to be more expensive than the task of recreating the design by black-box observation of its operation.

### A. Readback

From their inception, FPGAs of all types included a readback mechanism, whereby the program and data in the device can be read out for test purposes. To prevent unauthorized copy, early FPGAs followed the features of programmable logic devices (PLDs) and included a programming bit to disable the readback mechanism. This method worked well for antifuse and flash-based FPGAs, where the program could be loaded at a secure location, but SRAM FPGAs still needed to load the bitstream in the field, while potentially in the hands of an adversary. Preventing readback gave little protection if the bitstream could be intercepted as it was loaded into the FPGA. For this reason, antifuse FPGAs, that did not expose the programming in the system, gained an early reputation for being a more secure FPGA technology.

It is important to note that the readback function has, and continues to be, a valuable feature for both the FPGA manufacturer and the user. Whether the manufacturer uses it for device test, or the user employs readback for in-system data integrity checks, it is a feature, much like JTAG, that is useful but needs to be adequately protected to avoid vulnerabilities.

Readback continues to be a concern, and as late as 2012, Skorobogatov and Woods [38] discovered a keyed back-door/test mechanism that enabled the readback feature of a Microsemi antifuse FPGA that was assumed to be protected by the FuseLock protection mechanism [27].

### B. Early Bitstream Protections for SRAM FPGAs

Before bitstream encryption, two methods were used to protect SRAM bitstreams. The first method was to load the FPGA at a secure location and use a battery to hold the configuration bitstream for the entire lifetime of the fielded system [3]. Since programmable logic devices had privacy settings to prevent readback of the program, and since the bitstream was never exposed outside the device, this method assured that the bitstream running inside the FPGA is both secure and unmodified. This is precisely the same level of security achieved by antifuse and other nonvolatile FPGAs. The drawback of this method is, of course, the requirement that the system be powered continually. As FPGAs grew larger and more complex, this

solution became impractical due to increased standby power requirements.

The second solution was to use an external memory with a unique identifier, and customize the FPGA program to require that identifier, essentially tying the FPGA application bitstream to a unique board-level identifier. An improvement to the simple test of the board-level identifier uses an external keyed device that is queried with a random number generated by the FPGA [6]. This solution defeats simple cloning because the bitstream only functions correctly in the system with the correct external identifier device. However, the bitstream for each system is unique, which complicates the manufacturing process. Further, the design can be copied by an adversary who reverse engineers the bitstream, identifies the check logic, and rebuilds the application with the check removed.

This solution increases the difficulty and cost of copying the design but still relies on the difficulty of reverse engineering the bitstream as the basis of security. This solution was considered strong enough for many commercial applications, and there is no evidence that anyone mounted a successful attack on a device protected with it. However, reliance on the tedium and complexity of bitstream reverse engineering seemed risky [49].

### C. Modern FPGA Security

As FPGAs grew in capacity, the applications grew in value, driving the need for stronger security. Over the years, FPGA vendors have implemented circuitry, software, IP cores, and usage models to address security threats. Since the FPGA application design is embodied in a design file, aspects of information security, notably encryption and authentication, were applied to FPGA bitstreams. But that was not enough. Given that FPGAs were deployed into a hostile environment, measures were taken also to improve protocols and implementations to secure designs in the field. These include not only cryptography on the configuration files but also development of fault-tolerant design methodologies for the base array and for applications. Today, FPGA security is strong enough that they are deployed in security-sensitive applications in commercial and government systems [24].

## V. INFORMATION ASSURANCE

The basic tenets of information assurance (IA) are: confidentiality, integrity, availability, authentication, and nonrepudiation. As mentioned earlier, since access to the FPGA is assumed, availability is not a requirement addressed by FPGA security features. Nonrepudiation will be addressed in the context of authentication. Therefore, we focus on confidentiality, integrity, and authentication.

### A. Confidentiality

Large FPGA designs can contain IP of significant value, and bitstream encryption prevents a competitor from

simply copying that IP. Encryption can also provide trust assurance by limiting access to the FPGA only to designs constructed with the proper key.

1) *Overview of Bitstream Encryption*: Xilinx (San Jose, CA, USA) introduced bitstream encryption in 2001 in Virtex-II devices [40], [41] to address the problem of unauthorized copy of the bitstream as it is loaded into the FPGA from external memory. Since that time, other FPGA vendors have added encrypted-bitstream capability.

Preventing unauthorized copy does not strictly require encryption, since the task from a cryptographic point of view is to determine if the bitstream is authorized to operate in the FPGA. This fundamentally requires authentication, not confidentiality: a device could verify a message authentication code on the bitstream. However, the adversary's workaround is simple: reverse engineer the bitstream, recompile, and load it into a new FPGA with the authentication removed. Therefore, reverse engineering must also be prevented, so confidentiality of the bitstream becomes a requirement for preventing cloning.

Virtex-II FPGAs used triple-Data Encryption Standard (DES) encryption and subsequent Xilinx FPGAs use 256-b Advanced Encryption Standard (AES). Recent SRAM devices from Altera Corporation (San Jose, CA, USA) [4] and Flash devices from Microsemi Corporation (Aliso Viejo, CA, USA) [28] also use 256 b AES. Lattice Semiconductor (Hillsboro, OR, USA) devices use 128 b AES [19]. Although features have changed over the years, and details vary among vendors, the basics of FPGA bitstream encryption for all SRAM and Flash FPGAs are similar. The major components and use flow are described here with respect to the Xilinx, Inc. (San Jose, CA, USA) 7-series FPGA.

An application developer prepares a secured FPGA application with the same tools and processes used for any other application. At the end of the design process, when the bitstream is generated, Xilinx proprietary software encrypts the bitstream. The Xilinx software can supply a randomly generated key and initialization vector or the application developer may supply those values. The Xilinx software produces the encrypted bitstream and a key-insertion file.

2) *Key Loading*: At a secure facility, the application developer uses the key-insertion file to load the decryption key into the FPGA through the JTAG scan chain, as shown in Fig. 3. On-chip, the key is stored in either dedicated nonvolatile or volatile memory. FPGAs supply an independent battery-backed array for volatile storage or one-time-programmable eFuses for nonvolatile storage or both. Typically, the key is loaded into the FPGA in plaintext form, which is why this must be done at a trusted facility. Alternative strategies for key loading and key storage are discussed in Section VI-A2.

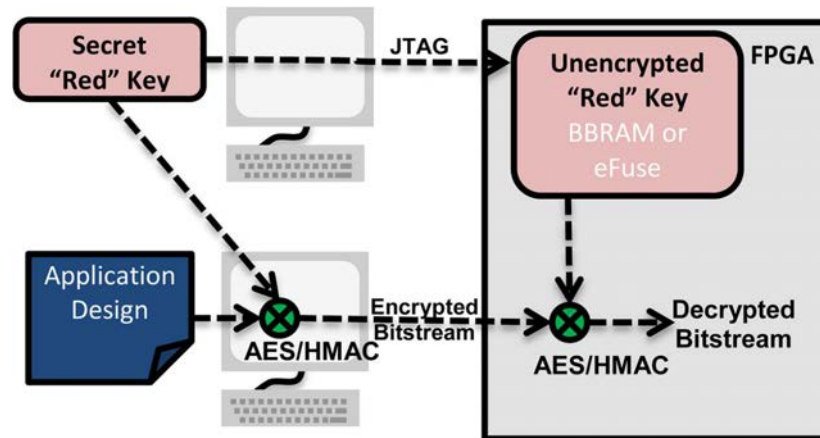


**Fig. 3.** Encryption architecture for Xilinx 7-series FPGAs.

3) *Bitstream Loading*: Later, in the field when the FPGA board boots, the FPGA loads its bitstream from an external memory. The FPGA begins loading an unencrypted bitstream. If the bitstream includes an encrypted-bitstream indicator, the FPGA starts the decryptor and decrypts the remainder of the bitstream as it loads. If the encrypted-bitstream indicator is not present, the FPGA bypasses the decryptor. This feature allows an FPGA in the field to be booted with either an encrypted bitstream or an unencrypted bitstream for test purposes without compromising the security of the bitstream confidentiality. In addition, most FPGAs now offer the ability to force the device to always configure with an encrypted bitstream.

## B. Data Integrity

Bitstream data integrity, the ability to ensure a design has not been accidentally modified, was a feature of very early FPGAs. In those early devices, an improperly programmed FPGA might enable two large internal drivers in contention, generating excessive heat and current, damaging the chip. To prevent this, data integrity checks were added to FPGA bitstreams to detect corruption of the bitstream during loading. Cyclic redundancy check (CRC), a common data integrity check in data transmission protocols, was deployed in many FPGAs. While CRC is effective in detecting accidental data corruption, it is ineffective against intentional data modification.

1) *Tampering With Encrypted Bitstreams*: Xilinx FPGAs use 256 b AES encryption [11] in cipher block chaining (CBC) mode of operation [33] to produce a stream cipher. In CBC encryption, each block of data is first xored with the ciphertext of the previous encryption before being encrypted. In decryption, the decrypted plaintext of each block is xored with the ciphertext of the previous block (Fig. 4). CBC causes blocks with identical plaintext (for example, all zero) to encrypt to different ciphertext,

thereby eliminating a dictionary attack on the data. Altera devices use AES in counter mode (CTR) [30]. In CTR mode, an encryptor encrypts the output of a counter to generate a pseudorandom stream of bits. That pseudorandom stream is xored with the plaintext to generate ciphertext. On decryption, an encryptor generates the same pseudorandom stream to recover the plaintext.

CBC and CTR are non-error-extension modes of operation, meaning that corruption of the encrypted data causes only a localized corruption of the corresponding plaintext. Therefore, both CBC and CTR permit a “bit-flipping” attack on the plaintext. The attack is shown in Fig. 4 with respect to CBC. If an adversary inverts a bit in the first encrypted block, as shown by the shaded area, the first block will decrypt to unintelligible nonsense. However, the corresponding plaintext bit in the next decrypted block is inverted. Bit-flipping CTR mode is more straightforward, since a bit flip anywhere in the ciphertext inverts the corresponding bit in the decrypted plaintext without disrupting any other data.

Using this bit-flipping technique, an adversary can selectively invert any number of bits in the decrypted bitstream. If the location and state of the target bit are known, an adversary can set it. For example, if the logic to enable bitstream readback is disabled with a “0” at a specific location, an adversary could reenforce bitstream readback without knowing the contents of the bitstream by inverting that one bit. For this reason, disabling of readback of an encrypted FPGA bitstream is not controlled by bits in the encrypted bitstream itself, but is instead controlled by the configuration logic of the FPGA. When the FPGA loads an encrypted bitstream, readback is disabled regardless of the bitstream contents. However, other attacks may attempt to modify the FPGA in a simple way: enable the internal configuration access port (ICAP), enable input/output (I/O) blocks, or change clock speed in an attempt to gain access to internal data.

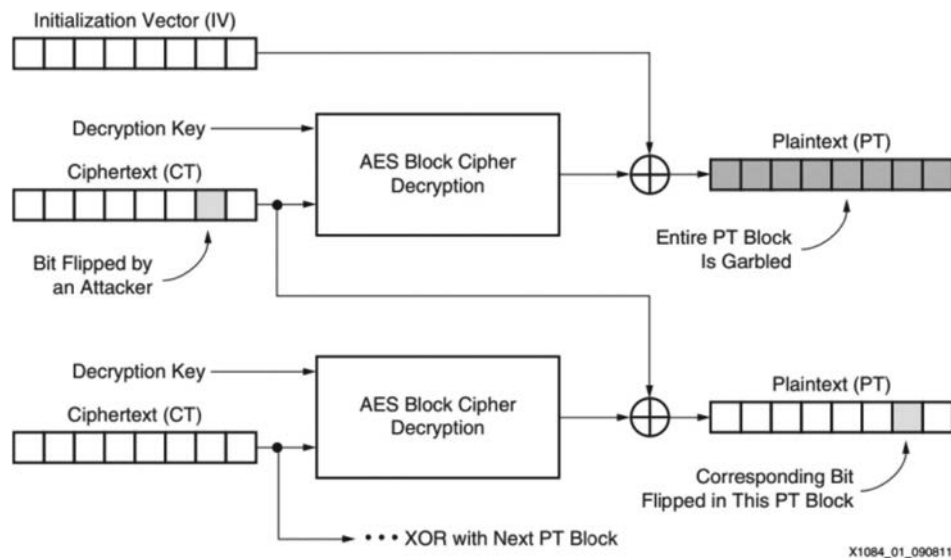


**Fig. 4.** Bit-flipping attack on CBC mode.

Attacks on the bitstream are also possible without knowing the specific bit to attack. In Fig. 4, the first block of data is scrambled. An adversary does not know the plaintext that results from modifying the ciphertext. If the number of bits to control is small enough, an attacker with patience may attempt a brute-force attack on part of the bitstream. Scrambling the bits may program the FPGA to perform a function it was not supposed to, such as leak-sensitive information.

Checksums and CRCs on the FPGA bitstream detect errors in transmission, corrupted bitstreams, and unintentionally flipped bits. However, it is computationally straightforward to compute a revised CRC after tampering with the bitstream or to determine a set of bit flips that produce the same CRC value. Further, a CRC is typically only 16 or 32 b, so brute-force attacks on the CRC are tractable. Finally, in some FPGA architectures, CRCs can be disabled altogether. Simple data integrity checks are not sufficient to ensure that a bitstream has not been intentionally tampered.

### C. Authentication

Communication of the bitstream to the FPGA is a one-way transfer. Therefore, two-way entity authentication cannot be performed. Instead, FPGAs rely on one-way message authentication, which assures the recipient of a message that the message is exactly the message the sender intended [8]. Strong authentication requires a message authentication code (MAC), a cryptographic hash function computed over the entire message. The hash function must be impossible to compute without knowing the plaintext of the message. The difficulty of recomputation of the MAC eliminates all forms of CRC as the hash function, since each bit of the CRC is a known XOR of a set of bits of the message.

Because authentication verifies that the application has not been accidentally or intentionally altered, it assures trust in the running application. That trust enables an application developer to guarantee protection of cryptographic services and the handling of sensitive data. These sensitive data may be customer data of high value, such as personal data in a database or copyrighted video. The cryptographic services may include key management functions, encryption/decryption algorithms, or keys for further partial reconfiguration of the FPGA. Data authentication provides a strong root of trust, allowing an initial FPGA configuration to act as a trusted boot loader for trusted subsequent configuration of the FPGA.

Xilinx integrated strong data authentication in Virtex-6 devices and 7-series to address the concerns of targeted tampering with encrypted bitstreams and the inherent cryptographic weaknesses of CRC. Microsemi also has a dedicated data integrity check for all of the nonvolatile configuration memory segments of some Flash devices [28]. Authentication is described here as it is implemented in Xilinx devices.

1) *Data Authentication in Xilinx Virtex Devices:* Virtex-6 and subsequent Xilinx FPGAs authenticate using the secure hash algorithm (SHA-256) to compute a 256-b keyed hashed MAC (HMAC) [12], [13], [42]. SHA-256 is a one-way hashing algorithm with a compact hardware implementation. The keyed HMAC requires a secret authentication key included in the hash. The MAC result cannot be computed without knowing the key, thereby authenticating the identity of the sender as well as verifying that the message has not been altered. The 256-b hash size ensures that any tampering with the bitstream will be detected with a high probability. HMAC with

SHA-256 makes tampering with the bitstream as computationally difficult as guessing the encryption key, which is also 256 b.

2) *Integration of Authentication With Bitstream Encryption*: Virtex devices use generic composition of the SHA-256 keyed HMAC authentication with AES-256 encryption [34], [37]. Generic composition allowed the two parts to be separated, which permitted them to be developed independently and separately pipelined.

Virtex-6 and 7-series authentication and encryption are composed using authentication then encryption (AtE). The HMAC is computed on the plaintext, unencrypted bitstream. The configuration data and the MAC result value are then encrypted. On the FPGA, the data are first decrypted and the MAC result is recomputed on the decrypted data and compared with the transmitted value in the bitstream. If the two MAC values disagree, the FPGA configuration fails and the FPGA does not become active. The authentication check catches errors in transmission and attempts to configure the FPGA with the incorrect key value as well as intentional tampering.

3) *The Authentication Key*: HMAC requires a secret authentication key in addition to the decryption key [12]. When generating an authenticated encrypted bitstream, both keys are specified to the bitstream generation software. To save nonvolatile storage space, only the decryption key is stored in the FPGA array. Because of the AtE composition, the encrypted authentication key can be transmitted with the bitstream. The bitstream encryption provides the privacy to keep the authentication key secret.

4) *Authentication Using Public Key Cryptography in FPGAs/SoCs*: The recent introduction of programmable systems on chip (SoCs) from FPGA manufacturers, including the Xilinx Zynq and Microsemi SmartFusion2 devices, have brought public key cryptography to the programmable logic market. Both of these devices use asymmetric cryptography to provide authentication during the secure boot process. The public key is stored on-chip in nonvolatile memory and its integrity checked before use. Public-key authentication of configuration files such as a first stage boot loader (FSBL) detects random-data attacks such as those commonly used for side-channel attacks. It can also serve to provide nonrepudiation of protected applications.

#### D. Bitstream Structure

Fig. 5 compares bitstream structures of representative Xilinx FPGA families, each with different security capabilities [42]. Fig. 5(a) shows the bitstream format of an unencrypted bitstream for Virtex devices. The unencrypted bitstream structure starts with a synchronization word (SYNC) followed by a sequence of instructions. Header commands set registers and control a variety of functions, including declaring the device type and setting

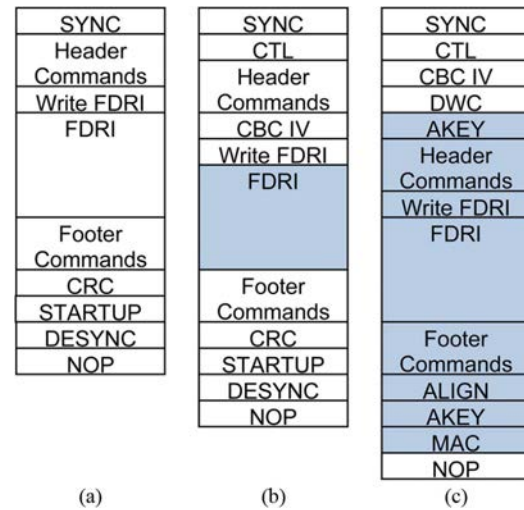


**Fig. 5. Xilinx bitstream structure. (a) Unencrypted. (b) Virtex-5. Shaded area is encrypted. (c) Virtex-6/7-series. Shaded area is authenticated and encrypted.**

up the startup sequence. The available commands and registers are described in the Configuration User Guides for each device family [50], [51]. The Write Frame Data Register Immediate (Write FDRI) command begins streaming the configuration data to the FPGA's configuration memory. An unencrypted bitstream can contain any number of Write FDRI commands, each writing a different, possibly discontinuous, portion of the FPGA configuration memory. Footer commands allow setting of register values after loading configuration data. CRC verifies data integrity and STARTUP begins the FPGA startup sequence. DESYNC prepares the configuration logic to accept postconfiguration reconfiguration commands.

Virtex-II through Virtex-5 FPGAs allowed encryption of the FPGA configuration data, but not authentication. As a representative of those encrypted-only bitstreams, Fig. 5(b) shows a Virtex-5 encrypted bitstream structure. The CTL instruction informs the FPGA that this is an encrypted bitstream. If the CTL command is missing, the FPGA assumes the bitstream is unencrypted. CBC IV is the initialization vector for the AES CBC register. The CBC IV does not need to be secret, and it is evident in the bitstream structure that it is set with an unencrypted header command. The Write FDRI command passes encrypted configuration data through the decryptor. The Write FDRI command includes a length field, also transmitted unencrypted, so the decryptor decrypts the proper amount of data. Only the configuration data are encrypted, although the CRC is computed on all data that precede it in the bitstream.

Fig. 5(c) shows an authenticated encrypted bitstream from Virtex-6 and 7-series devices. Authentication and encryption are always used together. There is no way to specify a bitstream that is only encrypted or only

authenticated for these devices. As in earlier devices, the CTL instruction informs the FPGA that the bitstream has security enabled. The CBC initialization vector initializes the decryptor as before. Decrypt word count (DWC) indicates to the FPGA the amount of secure data to follow. DWC includes not only the configuration data but header and footer commands as well. Header commands and footer commands are encrypted and covered by authentication. DWC is transmitted in the clear and could be modified by an adversary, but since the length of the data is included in the MAC computation, a modification of DWC will invalidate the computed MAC.

The authentication key is transmitted to the FPGA at the start of configuration and again at the end of configuration, since the key is used twice in the HMAC computation. ALIGN is a variable number of no-operation instructions, inserted to ensure the authenticated encrypted data are an even multiple of 512 b, simplifying the MAC computation. At the end of the bitstream, the required MAC is transmitted to the FPGA where it is compared with the MAC computed by the FPGA.

Confidentiality, data integrity, and data authentication of the configuration data are all required to protect FPGA configuration data that are exposed to potential adversaries. To date, only a few devices available from Xilinx and Microsemi provide all three protections on their configuration files.

## VI. ANTI-TAMPER

Physical security of the FPGA is just as critical as the application of confidentiality, integrity, and authentication to the device configuration. While there are focus areas of AT that overlap with IA, there are also aspects of AT that are unique. FPGA manufacturers are faced with a number of challenges while focusing on improving the physical security of the device. As commercial products, some of the challenges include, but are not limited to:

- FPGAs are readily available for adversaries to experiment on;
- compliance with U.S. and worldwide export and import restrictions—manufacturers must be able to sell their product worldwide, and do so while meeting all import/export laws;
- FPGAs are cost sensitive, requiring a careful balance between protecting customers' IP and enabling FPGA use in all types of systems.

There has been significant investment by FPGA manufacturers to enhance the physical security of their devices, driven primarily by the continual growth in performance, density, and capabilities. This puts FPGAs at the heart of most electronic systems today, where customers' IP must be protected.

This section describes some of the primary security features and protocols of the Xilinx 7-series FPGA. These are explored by looking at the configuration lifecycle of the

device. AT protections are employed preconfiguration, during configuration, and postconfiguration.

### A. Preconfiguration

1) *Defense Against Trojan Insertion*: FPGAs allow the ability to configure either encrypted or unencrypted. This is useful for application developers who may not want to use encryption during integration and test, but then enable encryption when the system is fielded. This ability to configure either encrypted or unencrypted, subjects the device to a class of Trojan insertion attacks.

If an FPGA contains a decrypted bitstream, an adversary may attempt to load a partial configuration into a subset of the device that spies on the resident application. It could connect to internal signals or memories. By connecting to internal components, the Trojan could be used to deduce the secured application in the FPGA. Conversely, an adversary may operate the same attack by preloading a Trojan design and interrupting the secure loading of the protected application.

Consequently, Xilinx FPGAs do not permit mixing encrypted and unencrypted bitstreams, or partial bitstreams, in any order. A new configuration of the device requires fully clearing the existing device, either by cycling power or executing the JTAG JPROGRAM command. Both methods initiate internal device housekeeping, which clears all configuration and internal memory.

Similar concerns exist today with SoCs being introduced by the FPGA manufacturers. Xilinx, Altera, and Microsemi are now offering processor-centric SoC devices that typically have separate and independent regions for the processor subsystem and the programmable logic. The independence provides users flexibility and the ability to significantly reduce power by turning off the programmable logic. This capability presents vulnerability. If an adversary can preload a Trojan, either into the processor memory or the programmable logic before allowing the device to boot normally, then the Trojan will have access to the entire internal application running on the device. As with Xilinx FPGAs, Xilinx SoCs have been designed to address this security concern. The Xilinx Zynq device boots both the processor and the programmable fabric from the same root of trust, either fully secured, or fully open.

The Trojan insertion vulnerability also exists postconfiguration. Xilinx and Altera FPGA families permit partial reconfiguration, the ability to change the configuration of a section of the FPGA while the rest operates normally. This feature has proven to be very valuable for innovative applications. However, it also is susceptible to a Trojan insertion attack after the initial configuration. The application design must authenticate postconfiguration bitstreams to exclude Trojans.

2) *Protecting Keys*: The secrecy of a cryptographic key is fundamental to security; protecting the key is the first

priority of the FPGA manufacturer. As stated earlier, this can be a challenge for commercial vendors who are developing a device that is used in nearly all types of applications, and not specifically designed for a specific domain, application, or cost point. Also important to note is the fact that while we address the protection of keys here in the preconfiguration section of this paper, protection of keys is essential before, during, and after configuration.

a) *Key storage—Technology*: Most FPGA manufacturers provide both volatile and nonvolatile key storage. In the case of SRAM FPGAs, volatile key storage is implemented as battery-backed RAM (BBRAM) and nonvolatile key storage is implemented as eFuses. Each has advantages and disadvantages.

BBRAM key storage requires no process changes, making it easily implemented in state-of-the-art process technology. The volatile key storage also allows for key agility and key zeroization, critical components of a strong cryptographic system. In Xilinx FPGAs, when primary power is applied, the BBRAM is powered by that power supply, which not only reduces the drain on the battery but also permits replacing the battery in fielded system. Altera specifies that a battery must be attached before the key is loaded, implying that the source for the BBRAM memory is only the external battery [4]. Xilinx also provides an internal interface that can be used by an application to command a zeroization of the key space. Zeroization is intended for use when the FPGA detects tampering with an operating application.

BBRAM is not without its disadvantages. A momentary loss of contact or low battery voltage could cause the key to be lost. While modern coin-cell batteries hold enough energy to hold encryption keys for the design lifetime of 20 years, and new betavoltaic batteries with great reliability are being introduced to the market, many battery vendors do not specify thermal wearout or other failure modes, for the length of time required by most FPGA users.

BBRAM is inherently more physically secure than nonvolatile key storage technology. To steal the key, an adversary would need to decap the FPGA and mill away many levels of metal, then scan the bits with a scanning electron microscope (SEM). This attack must be performed while keeping clean power to the key memory. This is the type of attack required to extract the entire configuration directly from the FPGA SRAM cells as well, so no bitstream encryption method is qualitatively stronger. This attack is considered to be beyond the capabilities of all but the most sophisticated of adversaries.

An eFuse provides a simple, one-time-programmable nonvolatile memory. Because they are nonvolatile, eFuses eliminate the maintenance issues associated with a battery. A common eFuse structure is a narrow wire that is programmed by electromigration from high programming current. eFuses are simple to build and program, requiring no additional process complexity or high voltage. However, eFuses and their programming circuitry are rather large, so

eFuses are practical only for small amounts of memory, such as a decryption key. The physical change caused by eFuse programming is visible under a microscope, so eFuses are comparatively easy to reverse engineer from a decapped part. Of course, they cannot be reprogrammed or erased. However, to zeroize an eFuse key, one could burn all eFuse cells in the key.

b) *Key loading*: The JTAG test port is a common interface for loading keys into programmable logic devices [4], [22], [35]. Loading a key into a Xilinx device begins by first executing a JTAG command to enter key access mode, which clears the existing key and all configuration data and memory in the FPGA. A second JTAG command writes the new key and reads it back to verify it. Of course, on power-up, FPGAs key access is disabled.

Details of key loading vary considerably among manufacturers. Loading of the key may be done in plaintext (“red key load”) or ciphertext (“black key load”) or otherwise obscured. In Xilinx devices, the key is transmitted to the FPGA in the plaintext, so it must be loaded in a secure location. The key access control sequence ensures that the key is cleared before any command is executed that could read it back. Other vendors have chosen alternative solutions. Altera Stratix devices include a key obfuscation mechanism so the key may be presented to the FPGA in an encrypted form. Moradi [32] reported that two 128 b keys are used by Altera Stratix-II devices. The bitstream key is transmitted and stored in encrypted form, encrypted by a second key, which is presented without any obfuscation. Although the key used to decrypt user data is not transmitted or stored in the FPGA, and hence cannot be extracted, it can be computed by a straightforward algorithm from the readable key that accompanies it. In Altera Stratix-V devices, the user key is sent through a one-way function before being stored on the device [4]. In both of these scenarios, the loading of the key is obscured, and while not cryptographically sound, may provide a level of security acceptable at a given price point.

Microsemi is the first FPGA manufacturer to offer a true “black-key load”: the key is encrypted by a secret key before loading. In selected SmartFusion2 devices, the device and user exchange public keys and perform an elliptic curve Diffie–Hellman (ECDH) exchange to generate a key that can be used for the authenticated/encrypted loading of a user key [28]. The generated key is used as a key encryption key (KEK) to encrypt the user key on the transmit side, and to decrypt the user key within the SmartFusion2 device.

c) *Key storage—red or black?*: Much like key loading, the device key can be stored in plaintext, ciphertext, or obfuscated form. Xilinx stores 7-series keys in plaintext form. An adversary who decaps the part and can identify the key storage cells can attempt to extract the actual key bits. An obfuscated key defeats this attack until the obfuscation method is discovered. Altera implemented a key obfuscation algorithm in Stratix devices, so that probing the device could not divulge the key directly. When the

obfuscation algorithm was revealed, obfuscation was no longer a barrier to invasive key extraction [32].

Selected SmartFusion2 devices from Microsemi make use of Intrinsic-ID's (Eindhoven, The Netherlands) Quiddikey technology [17]. This technology does not store an encryption key on-chip. Instead, it generates the key when needed through the use of an activation code generated during an enrollment phase, and the output of Intrinsic-ID's SRAM-based physically uncloneable function (PUF) [17], [29].

d) *Eliminating keys*: When an unauthorized event occurs, the application may need to eliminate sensitive keys within the device. For systems that employ BBRAM key storage, there are multiple options. First, passive erasure can be accomplished by simply electrically disconnecting the battery from the supply. Second, for Xilinx devices, an external device could send the appropriate JTAG command to enter key access mode. As mentioned earlier, this actively clears the device key and the configuration of the device. Finally, most FPGA vendors have the ability to erase the key from within the device under control of the application [4], [29], [38]. Xilinx and Microsemi offer the ability to fully zeroize the device key, actively erase the key, and then verify that it indeed has been erased, either through readback or dedicated hardware.

3) *Antispoofing*: When they are manufactured, FPGAs can accept either an unencrypted bitstream or an encrypted bitstream. All programmable logic vendors that provide encrypted bitstreams have the ability to modify the FPGA to require an encrypted configuration. This modification involves programming a nonvolatile eFuse register that disables unencrypted configuration. An adversary cannot substitute an alternative bitstream in the device or change the key. Instead, that adversary must replace the FPGA with another equivalent device. This solution provides no value with a BBRAM volatile key, as the adversary only needs to remove the battery to clear the key, then load a new key into the device to gain access. Of course, an adversary can circumvent the antispoofting by replacing the protected FPGA with a new, unprogrammed one. Nonetheless, antispoofting the device is a cost-effective component of overall system security.

4) *Test Circuitry*: Because it provides access to and control of internal nodes, test circuitry has long been a primary point of security vulnerability in integrated circuits, and must be disabled for a secure application to indeed be secure. While protection of test circuitry is discussed in this preconfiguration section, it must be considered during configuration and postconfiguration as well.

Test interfaces can be disabled in many ways. Proprietary test interfaces are typically handled differently than industry-standard interfaces such as JTAG. Xilinx disables readback by setting internal security bits when an encrypted bitstream is loaded. In the Zynq SoC, eFuses

may be used to disable test interfaces permanently [36]. Test disable is also provided in Altera devices where a tamper-protection bit disables the test modes of the FPGA [4]. When permanently disabling test circuitry, users must be aware of the consequences for additional failure analysis: if the test access port has been disabled, there is very little anyone can do to debug the device.

Microsemi and Xilinx provide mechanisms to permanently disable the JTAG interface as well as monitor it internally for tamper conditions [29], [35]. Altera has the ability to reduce the number of JTAG commands executed to only those mandatory by the standard (e.g., Extest, Intest, IDCODE, etc.). The execution of nonmandatory JTAG instructions can be enabled by issuing the UNLOCK JTAG instruction, which is only allowed to execute when sent from within the device [4].

## B. During Configuration

1) *Side-Channel Attacks on Keys*: In recent literature, Xilinx, Altera, and Microsemi FPGAs have been shown to be vulnerable to differential power analysis (DPA) attacks on their keys [30]–[32], [38]. Although noninvasive, these published attacks employ a custom board with a significant reduction in bypass capacitance in order to enhance the power signal. This brings up the question of the difficulty of moving an FPGA from one board to another while keeping the key intact. eFuse, antifuse, and Flash storage should be unaffected, but battery backed RAM keys are lost if, during the transfer, power is lost to the keys or if the device temperature exceeds operating limits.

Security is always a moving target. Attacks continue to improve, and since a custom board is not required, in principle, to mount a DPA attack, one would expect that future side-channel attacks on FPGAs will target devices in their native environment. Defenses improve as well. As side-channel attacks became better understood, FPGA vendors added countermeasures, though they are not always explicit about precisely what they have done. Microsemi has licensed CRI technology, but has not released which aspects of that technology they have used. Other vendors are silent on the question of precise circuit details to address DPA.

## C. Postconfiguration

FPGAs rely on the application as an active participant in protecting the device after configuration, a capability somewhat novel to FPGAs [45]. FPGAs provide security-related features, but leave the policy decision of handing the features to the user of the FPGA to implement in the application.

1) *Readback Disable*: Traditional FPGA operation allows the unencrypted bitstream and data to be read out using the bitstream readback command. Therefore, when an FPGA loads an encrypted bitstream, it disables the

readback mechanism, regardless of bitstream settings. This automatic, mandatory setting prevents the simple attack of using the FPGA to decrypt the bitstream, then reading it out. Readback continues to be a valuable feature for both the FPGA manufacturer and the application developer. Proper measures must be taken so that it does not jeopardize the application security.

Skorobogatov and Woods [38] used a side-channel attack to extract a key that unlocked readback in an FPGA that was advertised to have no such capability. While sensationalized as a back door, and questioned for who inserted it, and for whom, in all practicality it was no more than an interface used for device test.

2) *Restricted Access to Base Silicon Cryptographic Logic:* On-chip cryptographic functions, such as the decryptor, are well-tested, high-speed logic designs of a standard function. It would seem efficient to allow an operating FPGA application to use cryptographic functions after configuration. However, user access to the decryptor, or other cryptographic functions, permits data flow paths that complicate the analysis of the security of the base silicon. If the user has access to the cryptographic functions, and the device is programmed to permit unencrypted bitstreams, then the adversary has access to the cryptographic functions as well. The manufacturer must perform a security analysis to verify that no key data could leak into the application domain.

Second, there are U.S. export and various national import laws worldwide that add risk to the manufacturer if cryptographic functions are used for more than just the configuration of the device. Third, most cryptographic functions, such as AES decryptors are simply not very large and can be implemented in the user application without consuming much of the FPGA logic. Finally, users have a wide range of needs for cryptographic services. This becomes a cost/benefit tradeoff for the manufacturer. Xilinx and Altera do not allow access to the cryptographic functions on the FPGAs. Microsemi allows access to the cryptographic functions on selected models of the SmartFusion2 devices [28].

3) *Restricted Access to Base Silicon Features:* Concern over tampered bitstreams in early Virtex devices led Xilinx to prohibit reconfiguration of encrypted bitstreams. This restriction applied to the internal configuration access port (ICAP) as well as the external configuration port. The concern was that a bitstream might be tampered to enable access to ICAP, which could then be used to read back the decrypted configuration. Virtex-II through Virtex-5 devices required encrypted bitstreams to pass a CRC to begin operating, thus ensuring the integrity of the bitstream data. However, as described earlier, CRC does not give a strong defense against bitstream tampering.

Since the addition of authentication in Virtex-6 and 7-series, a secured bitstream must pass the authentication

check, defeating any bitstream tampering. Since an authenticated bitstream could not have been modified by an adversary, it can be trusted. This trust applies to the application in general, but specifically enables trusted self-reconfiguration with the ICAP. Since the application design is trusted, ICAP operation is permitted with authenticated encrypted bitstreams. An authenticated bitstream may use the ICAP to launch a partial configuration while the device continues to operate, allowing the design of a trusted reconfigurable platform [53].

ICAP is a Xilinx-specific example of a base silicon feature that, if used maliciously, could provide a vulnerability without the appropriate protections. In all cases, the manufacturer must provide safeguards, while the application developer has final responsibility. It is, of course, possible to construct an insecure application despite the encryption and authentication. For example, if an application developer connected the ICAP interface directly to the external pins, an adversary could interrogate the ICAP to read back the unencrypted application bitstream. FPGA security enables the construction of secure applications; it does not guarantee them.

4) *The Value of ICAP and Checking Designs in the Field:* ICAP permits logic inside the FPGA to read and write its own bitstream, providing a wide range of powerful use cases. These include:

- internal readback of the device configuration for in-system integrity checks;
- configuration clearing and zeroization;
- algorithm agility for those applications that need to change algorithms without a complete reconfiguration of the device;
- self-test;
- use of user-specific decryption and authentication algorithms with custom protections against attacks such as DPA or other side-channel attacks;
- configuration repair: random single-event upsets (SEU) [23], [52] or intentional tampering may cause configuration bits inside the FPGA to change. Jones [19] describes the SEU controller, an application in which the FPGA logic reads its own bitstream internally through ICAP, checks the stored bitstream with previously computed ECC data, and corrects configuration errors. The SEU controller is intended to detect and correct errors in a high-reliability environment, but it can be used to detect tampering with the FPGA in the field if individual bits are flipped. More recent FPGAs include the SEU detection and scrubbing feature in dedicated hardware [35].

## D. Invasive Attacks

Because of the environment of the fielded FPGA, the difficulty of protecting FPGA keys and configuration data persists regardless of the technology used to store them.

When an adversary can physically open the device and scan the contents, no storage technology is wholly secure. However, using our model of cost-based security, some storage methods are more expensive to break, sometimes despite having no qualitative advantage.

The strongest way to prevent theft and tampering with a bitstream is to keep it out of an adversary's hands. The Xilinx Spartan 3AN is a multichip package containing an FPGA die and a flash memory die. Since nothing is transmitted from an external source, the trivial bitstream interception method does not work. However, after decapping the package, the signals between die can be probed to pirate the bitstream.

MicroSemi's SmartFusion devices have internal non-volatile Flash memory storage as well. These devices are still subject to physical, invasive attack, though that attack is more difficult for several reasons. The storage in these devices is distributed around the device, and there is no localized point at which one could intercept the configuration data, so the attack must scan the entire device. Programmed Flash and antifuse cells are not observably changed from unprogrammed cells, so the detection of programming is more difficult. It may require SEM or thermal analysis. SEM images of programmed and unprogrammed antifuse cells show no apparent differences [2].

Invasive physical attacks on antifuse devices and Flash devices are qualitatively no more difficult than methods for extracting eFuse bits. However, these attacks are considered significantly more expensive because millions of bits of data must be extracted, rather than merely a 256-b key. Further, the resulting extracted programming is not formatted for programming another FPGA, so it must be formatted properly by the adversary in order to clone the design. The proper format is not published, so there is no cryptographically strong protection, but it is considered difficult and tedious.

Despite the concerns, there has not been a report of a successful invasive attack on any FPGA regardless of the internal storage: SRAM, BBRAM, eFuse, Antifuse, or Flash.

### E. Environmental Attacks

The circuits inside FPGAs that implement the security functions are no less susceptible to attack than those in other semiconductor integrated circuits. Published attacks on security functions in other devices include out-of-range temperature and power adjustment, overclocking, and other environmental attacks. Defense against these attacks is very difficult because, by definition, semiconductor foundries do not guarantee operation outside their guaranteed environmental range. FIPS140-2, level 4, requires environmental failure protection on cryptographic modules [10] and FPGA vendors provide limited protection from environmental attacks.

The traditional response to environmental attacks has been more robust circuitry, including dedicated voltage regulation for security functions, large hamming distances

in security-critical state machines, and redundant storage of critical state values, such as those disabling readback in a secure system.

Xilinx provides an embedded analog-to-digital converter (ADC) that can be used to monitor voltage and temperature both outside and inside the FPGA. Users can configure the circuitry to specific voltage and temperature ranges based on the environment the system will operate in. If the voltage or temperature exceeds this user-specified range, an internal alarm signal will be generated notifying the application running on the device. User-specific actions can then be taken, for example, clearing sensitive cryptographic variables in registers or RAMs, zeroizing the key or clearing the configuration of the device itself and shutting down.

With few exceptions, FPGA manufacturers do not publish details of their security circuitry. Microsemi FuseLock and FlashLock include internal fuses or flash cells that prevent inappropriate access. According to Microsemi, "special security keys are hidden throughout the fabric of the device, preventing internal probing and overwriting. They are located such that they cannot be accessed or bypassed without destroying the rest of the device" [28]. Xilinx readback disabling circuitry has "hardened triple-redundant logic" and key loading FSMs have "large hamming distances between states" [35].

1) *Device Identifier*: A unique identifier is a powerful way to restrict access to an FPGA, defeating cloning and spoofing. An application can be coded to operate only on the one device that matches a specific identifier or on a subset of devices with a range of values.

Modern FPGAs contain a device identification register. Xilinx provides device DNA, a 57-b serial number programmed in eFuses during manufacture and used for tracking devices. Device DNA is accessible from outside the FPGA via JTAG. In addition, devices include a user-programmable 32-b eFuse field that can be used as an identifier as well. This user eFuse field is only available to logic within the FPGA.

### F. PUFs and FPGAs

Other alternatives exist for device identifier. PUFs [14], [39] provide a device-specific unique identifier derived from random process variations. A PUF generator produces a different signature for each manufactured device. PUFs have been demonstrated in FPGA fabric ("soft PUF") as well as in dedicated logic ("hard PUF"). Microsemi's SmartFusion2 includes a hard PUF. Other FPGA vendors have IP providers who provide soft PUF functions in fabric. Therefore, application developers can build PUFs for device identification today with existing FPGAs.

There are several drawbacks for the use of PUFs that have precluded their use as decryption keys in FPGAs. First, the PUF only resides inside the device. It must be read out of the device to encrypt the bitstream data file.

Alternatively, Kean recommended an encryption method in which the FPGA encrypts its own data file using its internal key and emits the encrypted data for external storage [20]. More importantly, the PUF is unique to each unit, so the bitstream must be encrypted uniquely for each device. This problem may be addressed by including a key transformation word, the exclusive-or of the computed PUF for the device with the actual key used to decrypt the data. Still, at system build time, the FPGA must be powered on and the transformation word derived. Perhaps most importantly, PUFs are not stable: a few bits may change over the lifetime of the device. This is not particularly important for a device identifier, but disastrous for a decryption key. One method to compensate for this is the addition of helper data to the PUF-encrypted bitstream. Helper data are fundamentally error correcting code information for correcting errant bits in the PUF. It is unclear how much information about the key is leaked in helper data. Finally, long-term PUF reliability data over process, voltage, and temperature is sketchy at advanced process nodes, leading to concern over lost keys during the lifetime of the fielded device.

## VII. APPLICATIONS

### A. IFF Flow for Nonsecured Devices

Baetoni [6] described “identification friend or foe” (IFF), a way to tie an FPGA bitstream to a specific system. IFF uses an external storage device, a secure serial electrically erasable programmable read-only memory (EEPROM), such as the Dallas Semiconductor/Maxim DS2432 (Fig. 6). The secure EEPROM includes a cryptographic hash function. At system build time, the application developer programs a secret key into the EEPROM and also programs the secret key into the FPGA application.

After the FPGA boots, it uses its random number generator to interrogate the EEPROM. The EEPROM computes the hash of the random string with its stored key. The FPGA does the same. If the two hashes match, the FPGA continues to operate. If the hashes do not match, the

FPGA enacts countermeasures such as ceasing operation or disabling premium functionality. The check may be repeated as often as desired during operation.

IFF ties the FPGA bitstream to a properly programmed secure EEPROM. Although it can be applied to an FPGA without bitstream encryption, doing so leaves the system vulnerable. An adversary may reverse engineer the bitstream and disable the check on the hash function. This mechanism is even vulnerable with an encrypted, but not authenticated, bitstream, because an adversary may attempt to disable the hash function check by a bit-flipping attack or random perturbation of the plaintext to disable the hash check.

### B. Metered IP

As third-party IP cores become more common, one would like a mechanism to charge per copy for those cores. The core vendor would be paid for each use, just as if it had been a physical device. Guajardo *et al.* [15] described a method for doing this and the company Intrinsic-ID developed into a product under the brand name Quiddi-card [17].

The method has an enrollment phase and an authentication phase. In the enrollment phase, the FPGA is programmed with a PUF which generates an identifier unique to the FPGA. An activation code is generated from the PUF value and stored off-chip. The activation code generation is a proprietary algorithm, but may be an encryption of the PUF value using a private key of a public/private key pair. In the authentication phase, the same PUF is constructed in the FPGA and the design is authorized with the activation code (Fig. 7).

To turn this activation process into an IP metering mechanism, the generation of the activation code may be done by a trusted third party, possibly a trusted piece of billing hardware at a manufacturing site that reports the IP usage as it generates the activation code. This mechanism has been extended to include multiple keys to permit access to multiple pieces of IP in the FPGA application [18].

This mechanism relies on confidentiality and authentication of the application design, so that an adversary

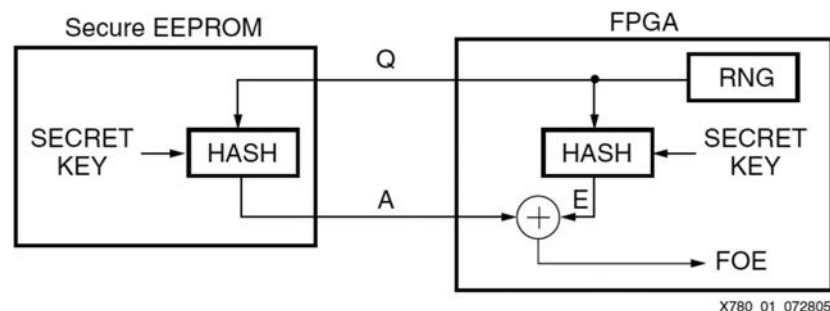


Fig. 6. IFF design.

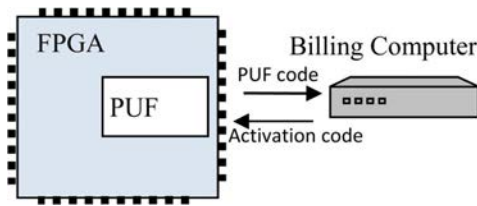


**Fig. 7. Metered IP system architecture.**

cannot reverse engineer the device to remove the activation code checking. There is nothing fundamental about using a PUF for identification. Device DNA or some other unique or nearly unique fixed device identifier can serve.

### C. Just in Time Secure Configuration

Utilizing partial reconfiguration and authenticated encrypted bitstreams, it is possible to design a system where critical technology (CT) is only configured into the device when it is needed, thereby adding an additional layer of security to the system. Peterson [35] proposed a method by which a user application is partitioned between CT and non-CT. The non-CT is resident in the FPGA at all times and the CT logic is partially reconfigured into the FPGA only when needed. Otherwise, it is stored externally, encrypted and authenticated.

The CT, which exists as a partial configuration, can be decrypted by the device using the device key or by the application using a user-specified algorithm implemented in the FPGA fabric, and potentially a PUF to generate the key. The boot configuration of the FPGA sends the CT partial bitstreams to the ICAP so that the decryption process is completely contained with the FPGA. Encryption is required to ensure the privacy of keys included in the CT partial bitstreams or the boot configuration bitstream. Authentication is required so that the bitstreams cannot be tampered in a way that compromises the CT partial bitstreams. The IP described by Zeineddini and Wesselkamper [53] for secure and high-reliability applications utilizing partial reconfiguration also checks for tampering. It uses the integrated ADC to monitor power and temperature, and checks the JTAG port to detect tamper conditions. If necessary, the IP zeroes the CT and its key.

### D. Fault-Tolerant Design

FPGA manufacturers supporting confidentiality, integrity, and authentication of the configuration provide a strong foundation that users can build high-reliability system upon. Cryptographic processing and security services, like any high-reliability function, must be fault tolerant. Xilinx's isolation design flow (IDF) [7], developed in conjunction with government entities [24], was the first in the programmable logic industry. Altera has since developed similar technology, called the design separation flow [5].

IDF provides fault containment at the FPGA module level, enabling single-chip fault tolerance by various techniques, including modular redundancy, watchdog alarms, segregation by safety level, and isolation of test logic for safe removal [7]. The applicability of this type of technology goes beyond cryptographic processing and security. The same technology can be used to aid in compliance for systems that must be designed to safety-critical standards such as IEC61508, ISO26262, and DO-254.

The basic concept is to separate critical and/or intentionally redundant functions physically on the FPGA. This can be accomplished through careful floorplanning and the use of unused logic as fences. Fig. 8 represents a design that has been floorplanned with IDF. Fig. 9 is the same design after place and route.

The fences are exhaustively analyzed by the FPGA manufacturer to show that a single failure would not compromise the isolation or redundancy built into the system. The goal is to minimize the size of the fence to reduce the inefficiencies that come with its use [16]. As an example, the width or height of a fence made of configurable logic blocks (CLBs) in a Xilinx 7-series FPGA is a single CLB.

In an ideal world, each module would be completely isolated from each other. In practice, this scenario is not feasible: some level of communication must exist between isolated regions. Xilinx developed the concept of "trusted routing," restricted routing that is specifically chosen by the place and route algorithms such that the isolation established by the use of "fences" is not compromised.

Finally, no high-reliability system is complete without the use of independent verification. To address concerns associated with software "bugs" or inappropriate use of the design methodology by the user, FPGA manufacturers must provide independent verification tools that can be applied to the design to validate the isolation of the modules. Xilinx

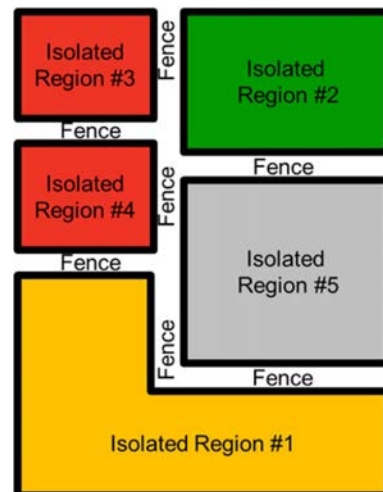


**Fig. 8. Notional floorplan of a design into five isolated regions.**

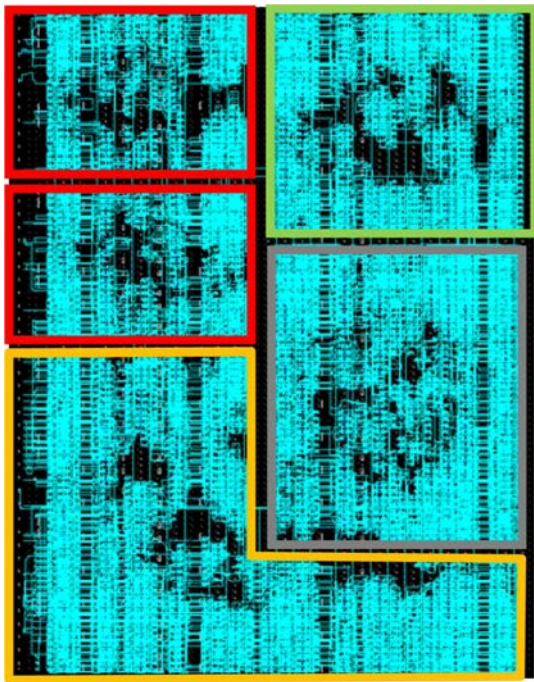


**Fig. 9.** FPGA editor view of a design implemented using the IDF methodology.

developed the isolation verification tool (IVT) for this purpose. IVT can be used early in the development flow to aid in isolation verification before a printed wiring board (PWB) is committed. It is also used once the design is complete in order to verify that the final design, placed and routed, has the isolation designed in that the user intended.

### E. Single-Chip Cryptography

Single-chip crypto (SCC) combines data of different levels of secrecy or control in a single device. The device must not only protect programs during loading, but also it must defend against attacks from outside and attacks while operating, including leakage of protected information across internal boundaries. Therefore, single-chip cryptography aggregates much of the technology discussed in this paper.

SCC uses the authenticated encryption capability to load a boot loader. The boot loader, isolation region #1 in Fig. 8, manages further FPGA configuration, software for on-chip processors, and data handling. Because it was authenticated and encrypted, the boot loader is known to be unaltered by potential adversaries or accidental bit errors. In addition, sensitive data, such as session keys, are known to be kept secret. To ensure no internal leakage of information, SCC implements the fences of IDF as described in Section VII-D (Fig. 8) to separate sensitive data spatially in the FPGA. This separation assures the confidentiality of sensitive information even in the presence of accidental or

intentional attacks on the fences. The spectrum of isolation capabilities is sufficient to support applications such as the separation of red and black data processing, key management, and other high-reliability functions.

Bitstream scrubbing, using internal readback, continually monitors the configuration data, in particular the isolation fences, to ensure that changes to the configuration are detected and corrected quickly. SCC can even verify that the device DNA is correct, ensuring operation on the proper individual chip.

Starting with the root of trust, followed by the power and flexibility of both hardware and software, coupled with the application of isolation technologies and partial reconfiguration, a system that would typically have been developed through the use of multiple devices now could be integrated into just one with no loss of security.

## VIII. THE FUTURE OF FPGA SECURITY

### A. Field-Programmable SoC

SCC was originally conceptualized and developed in cooperation with government authorities for FPGAs [24], and the application provides additional value in new programmable SoCs such as Zynq. Zynq includes both a programmable logic subsystem (PL) that comprises hundreds of thousands of gates of logic, and a processor subsystem (PS) that includes a dual-core ARM (ARM Holdings, Cambridge, U.K.) Cortex A9 processor, caches, memories, and peripherals, connected to one another and to the PL using an Advanced Microcontroller Bus Architecture (AMBA) Advanced eXtensible Interface (AXI) bus. The Zynq device boots securely, using authenticated encryption capabilities like those described for FPGAs. Zynq also provides asymmetric and symmetric authentication, confidentiality, and integrity. Leveraging this root of trust, applications can implement cryptoprocessors or systems performing cryptographic functions in the combination of processor and FPGA with confidence that they have not been compromised.

In Zynq, the processor subsystem is known to be isolated physically from the programmable logic. Within the PL, isolated regions as in IDF ensure separation of sensitive data spatially. Within the PS, known software methods, such as hypervisors and ARM Trustzone technology isolate sensitive software processes from other processes. The trusted boot loader decrypts and authenticates all configuration data and software.

Partial reconfiguration is further enhanced. The entire PL can be reconfigured, or even powered down, controlled by the PS. Alternatively, portions of the PL can be partially reconfigured for applications that require algorithm agility. The same reliability checks performed on ICAP [52] can be applied to the processor configuration access port (PCAP) to ensure proper data integrity of software. Decryption and authentication of partial configuration files

can be performed by either the PS or the PL, allowing users the flexibility to choose their own authentication and decryption algorithms as well as perform functions such as authenticate before decryption to aid in defense against side-channel attacks. Of course, key management remains a critical consideration in these applications.

## B. Conclusion

Security in FPGAs has been driven by the need to address new threats, by the growth in value of the IP of the applications, and by the growth in the expected sophistication of the adversary. All three drivers continue to operate. New areas of protection, such as confidentiality of the data handled by the FPGA, metering of third-party IP,

and counterfeit protection motivate additional capabilities and combinations of capabilities in the FPGA. Modern FPGAs and new programmable SoC devices hold applications that comprise complete systems, processing very sensitive data and controlling valuable systems. The high value of the applications, the data they handle, and the systems they control motivate well-equipped adversaries to steal IP or to subvert the systems of which the FPGA is a part.

As adversaries become more sophisticated, so do the FPGA defenses. Future FPGA security features must continue to improve to meet all three drivers. As in the past, these features will include circuits on the base array, algorithms in silicon, and IP in the programmable part of the device. ■

## REFERENCES

- [1] Actel, "Implementation of security in Actel's ProASIC and ProASICPLUS Flash-based FPGAs," Appl. Note AC185, 2003.
- [2] Actel, "Understanding Actel antifuse device security," 2004. [Online]. Available: [www.actel.com/documents/AntifuseSecurityWP.pdf](http://www.actel.com/documents/AntifuseSecurityWP.pdf)
- [3] P. Alfke, "Configuration issues: Power-up, volatility, security, battery back-up," Xilinx, Appl. Note XAPP092, 1997. [Online]. Available: [http://www.xilinx.com/support/documentation/application\\_notes/xapp092.pdf](http://www.xilinx.com/support/documentation/application_notes/xapp092.pdf)
- [4] Altera, "Using the design security features in Altera FPGAs," Appl. Note, AN-556, Jun. 19, 2013.
- [5] Altera, "Quartus II design separation flow," 2013. [Online]. Available: [http://www.altera.com/literature/hb/qts/qts\\_qii51019.pdf](http://www.altera.com/literature/hb/qts/qts_qii51019.pdf)
- [6] C. Baetoni, "FPGA IFF copy protection using Dallas Semiconductor/Maxim DS2432 Secure EEPROMs," Xilinx, Appl. Note XAPP780 v. 1.1, 2010. [Online]. Available: [http://www.zylinks.com/support/documentation/application\\_notes/xapp780.pdf](http://www.zylinks.com/support/documentation/application_notes/xapp780.pdf)
- [7] J. D. Corbett, "The Xilinx isolation design flow for fault-tolerant systems," Xilinx WP412, 2012. [Online]. Available: [http://www.xilinx.com/support/documentation/white\\_papers/wp412\\_IDF\\_for\\_Fault\\_Tolerant\\_Sys.pdf](http://www.xilinx.com/support/documentation/white_papers/wp412_IDF_for_Fault_Tolerant_Sys.pdf)
- [8] S. Drimer, "Authentication of FPGA bitstreams, why and how," *Reconfigurable Computing: Architectures, Tools and Applications*, vol. 4419. Berlin, Germany: Springer-Verlag, 2007, pp. 73–84.
- [9] S. Drimer, "Security for volatile FPGAs," Ph.D. dissertation, Comput. Sci. Dept., Cambridge Univ., Cambridge, U.K., 2009.
- [10] National Institute of Standards and Technology (NIST), "Security requirements for cryptographic modules," FIPS 140-2, 2001.
- [11] National Institute of Standards and Technology (NIST), "Announcing the advanced encryption standard," FIPS 197, 2001.
- [12] National Institute of Standards and Technology (NIST), "The keyed-hash message authentication code (HMAC)," FIPS PUB 198, Mar. 6, 2002. [Online]. Available: [http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1\\_final.pdf](http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf)
- [13] National Institute of Standards and Technology (NIST), "Secure hash standard," FIPS PUB 180-2 + Change Notice to include SHA-224, Aug. 1, 2002. [Online]. Available: <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenote.pdf>
- [14] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "Physical unclonable functions and public-key crypto for FPGA IP protection," *Proc. IEEE Int. Conf. Field-Programm. Logic Appl.*, 2007, pp. 189–195.
- [15] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "Brand and IP protection with physical unclonable functions," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2008, pp. 3186–3189.
- [16] T. Huffmire et al., "Moats and drawbridges: An isolation primitive for reconfigurable hardware based systems," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 281–295.
- [17] Intrinsic-ID, "Quiddikey-Flex," 2013. [Online]. Available: <http://www.intrinsic-id.com/products/quiddikey-flex>
- [18] Intrinsic-ID, "Quiddicard protecting your IP against overproduction, counterfeiting and cloning," Aug. 30, 2013. [Online]. Available: [www.intrinsic-id.com/products/quiddicard](http://www.intrinsic-id.com/products/quiddicard)
- [19] L. Jones, "Single event upset (SEU) detection and correction using Virtex-4 devices," Xilinx, Appl. Note #714, 2007. [Online]. Available: <http://www.xilinx.com/bvdocs/appnotes/xapp714.pdf>
- [20] T. Kean, "Secure configuration of field programmable gate arrays," in *Proc. IEEE Annu. Symp. Field-Programm. Custom Comput. Mach.*, 2001, pp. 259–260.
- [21] Lattice, "FPGA design security issues: Using Lattice FPGAs to achieve high design security," White Paper, 2007.
- [22] Lattice, "Advanced security encryption key programming guide for LatticeECP3, LatticeECP2MS, LatticeECP2S devices," Tech. Note TN1215, 2012.
- [23] A. Lesea, S. Drimer, J. Fabula, C. Carmichael, and P. Alfke, "The Rosetta experiment: Atmospheric soft error rate testing in differing technology FPGAs," *IEEE Trans. Device Mater. Reliab.*, vol. 5, no. 3, pp. 317–328, Sep. 2005.
- [24] M. McLean and J. Moore, "FPGA-based single chip cryptographic solution," *Military Embedded Systems*, 2007. [Online]. Available: <http://www.mil-embedded.com/pdfs/NSA.Mar07.pdf>
- [25] Microsemi, "Iglou2 FPGAs revision 0," 2013. [Online]. Available: [www.microsemi.com/document-portal/doc\\_download/132042-igloo2-fpga-datasheet](http://www.microsemi.com/document-portal/doc_download/132042-igloo2-fpga-datasheet)
- [26] Microsemi, "Axcelerator family FPGAs," 2012. [Online]. Available: [http://www.microsemi.com/document-portal/doc\\_download/130669-axcelerator-family-fpgas-datasheet](http://www.microsemi.com/document-portal/doc_download/130669-axcelerator-family-fpgas-datasheet)
- [27] Microsemi, "Implementation of security in Microsemi Antifuse FPGAs," Appl. Note AC168, 2012.
- [28] Microsemi, "Security architecture," 2013. [Online]. Available: <http://www.microsemi.com/products/fpga-soc/technology-solutions/security/security-architecture>
- [29] Microsemi, "SmartFusion2 SoC FPGA reliability and security user's guide," 2013.
- [30] A. Moradi, A. Barengi, T. Kasper, and C. Paar, "On the vulnerability of FPGA bitstream encryption against power analysis attacks: Extracting keys from Xilinx Virtex-II FPGAs," in *Proc. ACM Conf. Comput. Commun. Security*, 2011, pp. 111–124.
- [31] A. Moradi, M. Kasper, and C. Parr, "Black-box side-channel attacks highlight the importance of countermeasures—An analysis of the Xilinx Virtex 4 and Virtex-5 bitstream encryption mechanism," in *Proc. 12th Conf. Topics Cryptol.*, 2012, DOI: 10.1007/978-3-642-27954-6\_1.
- [32] A. Moradi, D. Oswald, C. Paar, and P. Swierczynski, "Side channel attacks on the bitstream encryption mechanism of Altera Stratix II," in *Proc. ACM/SIGDA Int. Symp. Field-Programm. Gate Arrays*, 2013, pp. 91–100.
- [33] National Institute of Standards and Technology (NIST), "Recommendation for block cipher modes of operation," Special Publ. 800-38A, 2001.
- [34] M. Parlekar, "Authenticated encryption in hardware," M.S. thesis, Electr. Comput. Eng. Dept., George Mason Univ., Fairfax, VA, USA, 2005.
- [35] E. Peterson, "Developing tamper resistant designs with Xilinx Virtex-6 and 7 series FPGAs," Xilinx, Appl. Note XAPP1084, 2012.
- [36] L. Sanders, "Secure boot of Zynq-7000 all-programmable SoC," Xilinx, Appl. Note XAPP 1175 (v1.0), 2013.
- [37] B. Schneier, *Applied Cryptography Second Edition*. New York, NY, USA: Wiley, 1996.
- [38] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," *Cryptographic Hardware and Embedded Systems—CHES 2012*, vol. 7428. Berlin, Germany: Springer-Verlag, 2012, pp. 23–40.
- [39] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. Design Autom. Conf.*, 2007, pp. 9–14.

- [40] A. Telikepalli, "Is your design secure?" Xilinx, 2003. [Online]. Available: <http://www.xilinx.com/publications/archives/xcell/Xcell47.pdf>
- [41] S. Trimberger, "Method and apparatus for protecting proprietary configuration data for programmable logic devices," U.S. Patent 6 654 889, 2003.
- [42] S. Trimberger, J. Moore, and W. Lu, "Authenticated encryption of FPGA bitstreams," in *Proc. 19th ACM/SIGDA Int. Symp. Field Programm. Gate Arrays*, 2011, pp. 83–86.
- [43] S. Trimberger, *Field-Programmable Gate Array Technology*. Norwell, MA, USA: Kluwer, 1994.
- [44] S. Trimberger, "Trusted design in FPGAs," in *Proc. Design Autom. Conf.*, 2007, pp. 5–8.
- [45] S. Trimberger and J. Moore, "FPGA security: From features to capabilities to trusted systems," in *Proc. 51st Annu. Design Autom. Conf.*, 2014, DOI: 10.1145/2593069.2602555.
- [46] S. Trimberger, "Security in SRAM FPGAs," *IEEE Design Test Comput.*, vol. 24, no. 6, p. 581, Nov./Dec. 2007.
- [47] S. Trimberger, "Three ages of FPGAs," in *FPGA20. Highlights of the International Symposium on Field-Programmable Gate Arrays*, ACM, 2011, pp. 1–18.
- [48] T. Tuan, T. Strader, and S. Trimberger, "Analysis of data remanence in a 90 nm FPGA," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2007, pp. 93–96.
- [49] T. Wollinger and C. Parr, "How secure are FPGAs in cryptographic applications," *Field Programmable Logic and Application*, vol. 2778, P. Y. K. Cheung, G. A. Constantinides, and J. T. de Sousa, Eds. Berlin, Germany: Springer-Verlag, 2003, pp. 91–100.
- [50] Xilinx, "Virtex-4 FPGA configuration user guide, v1.11," UG071, 2009.
- [51] Xilinx, "Virtex-6 FPGA Configuration User Guide," UG360, Jul. 30, 2010. [Online]. Available: [http://www.xilinx.com/support/documentation/user\\_guides/ug360.pdf](http://www.xilinx.com/support/documentation/user_guides/ug360.pdf)
- [52] Xilinx, "Device reliability report, second quarter 2013," UG116, 2013.
- [53] A. Zeineddini and J. Wesselkamper, "PRC/EPRC: Data integrity and security controller for partial reconfiguration," *Appl. Note XAPP887*, 2012.

## ABOUT THE AUTHORS

**Stephen M. Trimberger** (Fellow, IEEE) received the B.S. degree in engineering and applied science from the California Institute of Technology, Pasadena, CA, USA, in 1977, the M.S. degree in information and computer science from the University of California at Irvine, Irvine, CA, USA, in 1979, and the Ph.D. degree in computer science from the California Institute of Technology in 1983.

He was employed at VLSI Technology from 1982 to 1988. Since 1988 he has been at Xilinx, San Jose, CA, holding a number of positions. He is currently a Xilinx Fellow, heading the Circuits and Architectures group in Xilinx Research Labs in San Jose, CA, USA. He is an author and editor of five books as well as dozens of papers and journal articles. He is an inventor on more than 200 U.S. patents in the areas of integrated circuit (IC) design, field-programmable gate array (FPGA) and application-specific integrated circuit (ASIC) architecture, computer-aided engineering (CAE), 3-D die stacking semiconductors, and cryptography.

Dr. Trimberger is a four-time winner of the Freeman Award, Xilinx's annual award for technical innovation. He is a Fellow of the Association for Computing Machinery (ACM).



**Jason J. Moore** received the B.S. degree in electrical engineering from New Mexico State University, Las Cruces, NM, USA, in 1992.

He is currently a Director of Market Segments Engineering at Xilinx, Albuquerque, NM, USA, focused on security and safety architectures. Previous to his assignments at Xilinx, he was responsible for the development of field-programmable gate array (FPGA)-based communication security equipment in a wide range of avionics and ground-based platforms at the Motorola Government Group. He has been awarded multiple patents on cryptographic design in addition to novel approaches for logical and functional isolation within a single FPGA.

Mr. Moore is a two-time winner of the Freeman Award, Xilinx's annual award for technical innovation.



# Regaining Trust in VLSI Design: Design-for-Trust Techniques

*This paper surveys how concepts in VLSI test can be adopted in the context of hardware security and trust and design-for-trust techniques.*

By JEYAVIJAYAN RAJENDRAN, *Student Member IEEE*, OZGUR SINANOGLU, *Member IEEE*, AND RAMESH KARRI, *Senior Member IEEE*

**ABSTRACT** | Designers use third-party intellectual property (IP) cores and outsource various steps in their integrated circuit (IC) design flow, including fabrication. As a result, security vulnerabilities have been emerging, forcing IC designers and end-users to reevaluate their trust in hardware. If an attacker gets hold of an unprotected design, attacks such as reverse engineering, insertion of malicious circuits, and IP piracy are possible. In this paper, we shed light on the vulnerabilities in very large scale integration (VLSI) design and fabrication flow, and survey design-for-trust (DfTr) techniques that aim at regaining trust in IC design. We elaborate on four DfTr techniques: logic encryption, split manufacturing, IC camouflaging, and Trojan activation. These techniques have been developed by reusing VLSI test principles.

**KEYWORDS** | Design automation; design for testability; security

## I. INTRODUCTION

Typically, an integrated circuit (IC) comprises a wide variety of components, including from digital, analog, photonic to microfluidic [1]. Sensors, actuators, and biological interfaces are also being integrated into these ICs. On the

one hand, IC designs have been enabled by advances in mixed system integration and the increase in the wafer sizes (currently about 300 mm and projected to be 450 mm by 2018 [1]), reducing the cost per IC. On the other hand, support for multiple capabilities and mixed technologies has increased the cost of owning an advanced foundry. For instance, the cost of owning a foundry will be \$5 billion in 2015 [2]. Consequently, only high-end commercial foundries now manufacture high-performance, mixed system ICs, especially at the advanced technology nodes [3]. Absent the economies of scale, many of the design companies cannot afford owning and acquiring expensive foundries, and hence, outsource their fabrication process to these “one-stop-shop” foundries.<sup>1</sup>

While such globalization of IC design flow has successfully ameliorated the design complexity and fabrication cost problems, it has led to several security vulnerabilities [4]. If a design is fabricated in a foundry that is not controlled by the (fabless) design house, reverse engineering, malicious circuit insertion/modification, and intellectual property (IP) piracy are possible [3]. A rogue element in the foundry or a malicious user can reverse engineer the functionality of an IC/IP, and then steal and claim ownership of the IP [5]. An untrusted IC foundry may overbuild ICs and illegally sell these excess parts [6], [7]. Rogue elements in the foundry may insert malicious circuits (hardware Trojans) into the design unbeknownst to the designer [8], [9]. Finally, an attacker can reverse engineer an IC by depackaging and delayering it, imaging the individual layers, stitching the images, and extracting the netlist. This way, he/she can steal an IP or reveal competitor’s trade secrets. Because of these attacks, the semiconductor industry loses several billions of dollars annually [10]. The underlying reason is that the designers

<sup>1</sup>Companies that do not own their foundry are termed “fabless” design houses.

Manuscript received October 25, 2013; revised May 30, 2014; accepted June 14, 2014. Date of publication July 15, 2014; date of current version July 18, 2014. This work was supported in part by the Semiconductor Research Corporation and Advanced Technology Investment Company (SRC-ATIC) under Grant 2013-HJ-2440-S4; the New York University/New York University Abu Dhabi (NYU/NYU-AD) Center for Research in Information Security Studies and Privacy (CRISSP); the National Science Foundation, Computing and Communication Foundations (NSF-CCF) under Grant 1319841; the NSF-CI-ADDO-NEW under Grant 1059328; and the U.S. Army Research Office (ARO) under Grant W911NF-13-1-0272.

**J. Rajendran** and **R. Karri** are with the Electrical and Computer Engineering Department, Polytechnic School of Engineering, New York University, Brooklyn, NY 11201 USA (e-mail: jv.ece@nyu.edu; rkarri@nyu.edu).

**O. Sinanoglu** is with the New York University-Abu Dhabi, Abu Dhabi 129188, United Arab Emirates (e-mail: os22@nyu.edu).

Digital Object Identifier: 10.1109/JPROC.2014.2332154

0018-9219 © 2014 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See [http://www.ieee.org/publications\\_standards/publications/rights/index.html](http://www.ieee.org/publications_standards/publications/rights/index.html) for more information.

have minimum control over their design in this distributed, and potentially, vulnerable design and fabrication flow.

While hardware security and trust is a relatively recent but an important problem, a somewhat similar yet fundamentally different problem of manufacturing defects has been extensively researched by very large scale integration (VLSI) test researchers for the last few decades. We leverage the knowledge gained from VLSI testing and apply it to hardware security. On the one hand, while manufacturing defects are natural and unintentional, the attacks outlined above are man-made, intentional, and meant to be hidden. On the other hand, many concepts in VLSI testing, such as justification and sensitization (see Section II), can be adapted in the context of hardware security. Inspired by the design-for-testability (DfT) solutions for better testability of manufacturing defects, one can develop design-for-trust (DfTr) solutions to detect and prevent these attacks.

In this paper, we shed light on the vulnerabilities in the VLSI design and fabrication flow and survey four DfTr techniques. These DfTr techniques are as follows.

- 1) *Logic encryption* implements a built-in locking mechanism on ICs to prevent reverse engineering and IP piracy by a malicious foundry and user, and hinder Trojan insertion by a malicious foundry.
- 2) *Split manufacturing* splits the layout and manufactures different metal layers in two separate foundries to prevent reverse engineering and piracy by a malicious foundry.
- 3) *IC camouflaging* modifies the layout of certain gates to deceive reverse engineers into obtaining an incorrect netlist, thereby, preventing reverse engineering by a malicious user.
- 4) *Trojan activation* inserts dummy scan flip-flops (dSFFs) to manipulate the transition probabilities and expose a Trojan that is hidden in low-activity regions.

When these techniques are applied arbitrarily with no consideration of the design structure, they fail to provide the required levels of security. Consequently, they are easy to break and provide confidential information to the attacker. We also show that by defining appropriate security metrics and incorporating techniques such as fault activation, sensitization, and masking, these DfTr techniques can be made stronger, providing the security levels needed to regain trust in hardware.

The paper is organized as follows. Section II describes the VLSI test principles. Section III elaborates on the four DfTr techniques, their security requirements, and the shortcomings of the naive approaches. Section IV explains how VLSI testing principles can help these DfTr techniques in meeting their security requirements. Section V provides a discussion on the DfTr techniques. Section VI concludes the paper.

## II. BACKGROUND: TEST PRINCIPLES AND THEIR APPLICATIONS

In this section, we describe the VLSI test principles that we will use to develop DfTr techniques [14].

- Test principle 1 (fault excitation): A stuck-at- $v$  fault at a site is excited when an input pattern justifies that site to  $\bar{v}$ .
- Test principle 2 (sensitization): A site is sensitized to an output if every side input of every gate on a path from the site to the output is justified to the noncontrolling value of the gate. Sensitization of an internal line  $l$  to an output  $O$  refers to the condition (values applied from the primary inputs to justify the side input of gates on the path from  $l$  to  $O$  to the noncontrollable values of the gates) which bijectively maps  $l$  to  $O$ , and thus, renders any change on  $l$  observable on  $O$ .
- Test principle 3 (fault propagation): The effect of a fault at a site propagates to an output if the input pattern excites the fault and sensitizes the faulty site to the output.
- Test principle 4 (fault masking): Multiple effects of the same excited fault or multiple excited faults mask each other when none of their effects manifest at the outputs, as the errors cancel out.
- Test principle 5 (controllability): It is a measure of difficulty of setting a wire to a desired value (logic 1 or logic 0).

## III. BASELINE DfTr TECHNIQUES

In this section, we focus on four DfTr techniques: logic encryption, split manufacturing, IC camouflaging, and Trojan activation. For each of these techniques, we describe the threat model and different design approaches proposed in the literature, and introduce their security criteria and metrics. In addition, for each technique, we explain why naive approaches fail to meet the criteria, motivating VLSI-testing-inspired DfTr approaches.

### A. DfTr1: Logic Encryption

Logic encryption hides the functionality and the implementation of a design by inserting additional gates, referred to as key gates, into the original design. In order for the encrypted design to exhibit its correct functionality (i.e., produce correct outputs), a valid key has to be supplied to the encrypted design (for example, loading the key to a tamper-proof on-chip memory [15]). Upon applying an incorrect key, the encrypted design will exhibit an incorrect functionality (i.e., produce incorrect outputs).

Logic encryption of hardware does not mean encrypting the design file by a cryptographic algorithm; instead, it means encrypting the hardware's functionality. Researchers have previously used the term "logic obfuscation" [6],

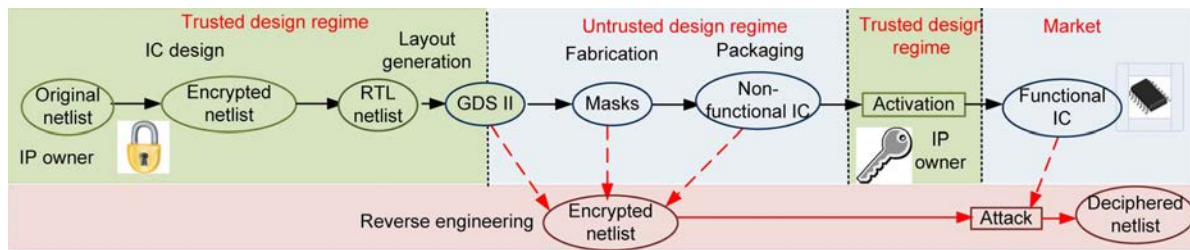


**Fig. 1.** Logic encryption in an IC design flow [7], [11]–[13]. The top part depicts an IC design flow enhanced with logic encryption capabilities to thwart IP piracy, IC overproduction, reverse engineering, and Trojan insertion [6], [7]. Before sending the design to an untrusted foundry, the designer encrypts the design using logic encryption techniques. The foundry then manufactures this encrypted design. On receiving the encrypted hardware, the IC designer activates it by applying the secret key and the IC is then sold in the market. The bottom part depicts the threat model for logic encryption. In the untrusted regime, an attacker can obtain the encrypted netlist from the IC design, or by reverse engineering the layout, mask, or a fabricated IC, and the functional IC from the market. Using this attack, the attacker can get a deciphered netlist and make pirated copies, overproduce ICs, or insert Trojans at “safe” places.

[7] for this purpose. Obfuscation of a hardware hides only its functionality, but obfuscation does not prevent black-box usage (obtaining an output from a hardware on applying an input) [16]. Logic encryption prevents this black-box usage of the encrypted hardware in addition to hiding its implementation. Hence, we use the term “encryption,” and not “obfuscation.”

1) *Threat Model*: Logic encryption techniques can thwart an untrusted foundry from illegally copying, reverse engineering, and overproducing the IC design [6], [7]. Fig. 1 shows the threat model for logic encryption. The IP provider and the designer are trusted. The attacker is in the foundry. The designer encrypts the (security-critical) modules using the proposed technique, synthesizes them using trustworthy computer-aided design tools, and sends the generated layouts to the untrustworthy foundry. The foundry manufactures the encrypted IC and returns them to the designer. The designer then applies the secret key and makes the ICs functional. The designer or a trusted third-party performs functional validation and manufacturing testing on this functional IC. Once the ICs pass the tests, the functional ICs are packaged and sold. The secret keys for an IC are stored in a tamper-proof on-chip memory, and thus, are assumed to be unrecoverable by a malicious user.

We consider two types of attackers: a rogue element in the foundry and a malicious user. In the first type, the rogue element in the foundry has access to the layout files (in standard GDSII format [2]) given to him/her for manufacturing the IC. This attacker can pirate the IP and/or overproduce IC. In addition, this attacker can analyze the design to identify the structural behavior of the design, thereby finding safe places to insert Trojans. In the second type, a malicious user can buy the IC, depackage and delayer it, image the individual layers, stitch the images, and extract the gate-level netlist. This attacker can reverse engineer an IC and pirate the IP.

Logic encryption prevents these attacks by encrypting all or critical modules in a design. Since the design is encrypted by the designer, without the secret keys, the foundry cannot use any copies or overproduced ICs. Furthermore, it prevents an attacker from analyzing the structural behavior of the design, thereby hindering Trojan insertion. Additionally, if the target IC is sold in the market, an attacker can buy one of the functional ICs. In this way, an attacker can gain access to good input/output (I/O) pairs. Furthermore, the attacker can identify the inputs of key gates through structural analysis of the netlist.

2) *Previous Work*: Logic encryption can be broadly classified into two types: sequential and combinational. In sequential logic encryption, additional logic (black) states are introduced in the state transition graph [7], [11], [12]. The state transition graph is modified in such a way that the design reaches a valid state only on applying a correct sequence of key bits. If the key is withdrawn, the design, once again, ends up in a black state.

In combinational logic encryption, XOR/XNOR gates are introduced at random locations in the design to conceal its functionality [6]. One of the inputs of these inserted gates serves as the key input. One can configure these inserted gates as buffers or inverters using these key inputs. The values applied to these key inputs are the keys. To generate unique keys per IC, [17] calibrates the XOR/XNOR gates postfabrication by tweaking the threshold voltage of the gates and introducing process variation sensors into a circuit. To make the IC functional, these sensors have to be configured by applying the secret key. Instead of XOR and XNOR gates, one can also use multiplexers as key gates [18]. The select line of the multiplexers is connected to the key inputs.

Memory elements may also be inserted into the design [19]. The circuit will function correctly only when the memory elements are programmed correctly. One can

consider these programming bits as the secret keys, similar to that of logic encryption. However, the insertion of memory elements may incur significant power, area, and delay overheads.

3) *Criteria*: A logic encryption technique should satisfy two criteria: a) incorrect outputs should be produced on applying an incorrect key; and b) an attacker should not be able to retrieve the secret key.

*Criterion 1 (50% Output Corruption)*: The objective of the defender (designer) is to prevent his/her IP from being copied by an attacker in the foundry and to prevent black-box usage. The attacker does not know the secret key used for encryption. Hence, he/she will apply a random key and in turn expect the module to become functional (i.e., to produce correct outputs). In the worst case, he/she has to apply all possible keys. If the key size is sufficiently large (say 128), then the number of possible key combinations will be large ( $2^{128}$ ). Consequently, the brute force attack becomes computationally harder. Thus, the objective of the defender is to force the attacker to perform a brute force attack. To achieve this objective, the defender has to encrypt the module such that an attacker, with the knowledge of the publicly available logic encryption objectives and algorithms, is not able to obtain the correct outputs by applying an incorrect key. This can be done by minimizing the correlation between the corrupted and original outputs, thereby maximizing the ambiguity for the attacker.

The ambiguity for an attacker is maximum when 50% of the outputs are corrupted upon applying a random incorrect key (a mathematical derivation for this metric is provided in [18]). One can quantify this metric using the Hamming distance. The Hamming distance between the correct output and the output on applying a random incorrect key should be 50% on average. The Hamming distance is defined as the number of output bits that differ on applying a valid key versus an invalid key for the same input. Ideally, 50% of the output bits should differ on applying any invalid key for any input. In the case of logic encryption, the Hamming distance metric is formally defined as follows. For a circuit  $C$  which produces an output  $y$  for an input  $x$  on applying the valid key, logic encryption of  $C$  should satisfy

$$\frac{\sum_{x \in X} \sum_{k \in K; k \neq \text{valid key}} HD(y_{\text{valid key}, x}, y_{k, x})}{2^{|x|} \times (2^{|k|} - 1) \times |y|} \times 100\% = 50\% \quad (1)$$

where  $X$  is the set of all inputs,  $K$  is the set of all possible keys, and  $y(k, x)$  is the output vector on applying the key  $k$  for the input  $x$ .  $|x|$ ,  $|y|$ , and  $|k|$  denote the size of the input, output, and key in bits, respectively.

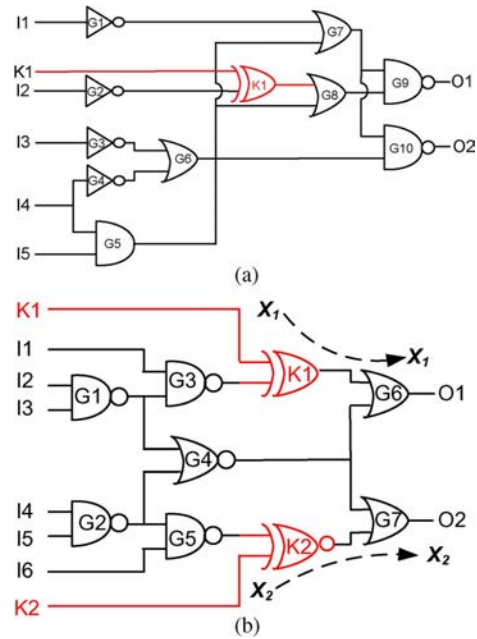


**Fig. 2. Problems with naive logic encryption. (a) Circuit produces correct outputs for 20 out of the 32 input patterns even on applying an incorrect key to the key gate (K1). (b) An attacker can observe the key bits at the outputs by applying the input pattern 100000 [13], [20].**

*Problem 1*: Random insertion of key gates does not ensure an incorrect output is produced for an incorrect key, let alone 50% output corruption.

*Example 1*: Let us consider the combinational logic encryption technique proposed in [6] and [7]. In this technique, XOR/XNOR gates are inserted at random locations. For instance, consider the C17 circuit shown in Fig. 2(a) encrypted with one XOR gate K1. The design will produce the correct output on applying the correct key value  $K1 = 0$ . On applying an incorrect key ( $K1 = 1$ ), one expects incorrect outputs. For example, on applying the input pattern 01000, an incorrect output 00 is produced instead of the correct output 10. Unfortunately, the design produces correct outputs for most of the input patterns even on applying an incorrect key. For example, the input pattern 11100 produces the correct output 11 even when an incorrect key is applied. In fact, this design produces an incorrect output only for 12 out of the possible 32 input patterns. In other words, the design produces correct outputs for 62.5% (20) of the input patterns despite applying the incorrect key. Thus, this encryption procedure is weak as it does not ensure that incorrect outputs are produced for incorrect keys, let alone the 50% output corruption criterion.

*Criterion 2 (Difficult-to-Break Encryption)*: A logic encryption should ensure the secrecy of the key even when an attacker has access to good I/O pairs.

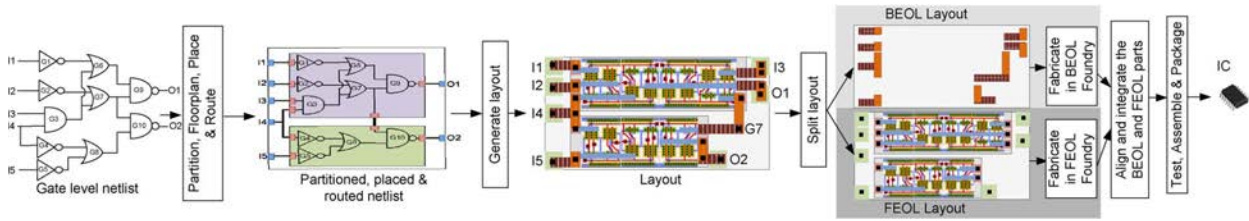


**Fig. 3. Split-manufacturing-aware IC design flow. The layout obtained from the traditional design flow is split into two parts: the FEOL part containing the transistors and the lower metal layers, and the BEOL part containing the top metal layers. These parts are then fabricated in two different foundries. Either the designer or the BEOL foundry assembles the FEOL and BEOL wafers into the final IC [3], [21].**

*Problem 2:* Random insertion of key gates fails to ensure the secrecy of keys.

*Example 2:* Consider the encrypted circuit shown in Fig. 2(b). The key gates K1 and K2 are randomly inserted in the circuit. By applying the input pattern 100000, an attacker can observe the secret values of the key bits at the outputs O1 and O2. Thus, random encryption fails to ensure the secrecy of key bits.

## B. DfTr 2: Split Manufacturing

Split manufacturing has been proposed to thwart a foundry from accessing the entire design information and from stealing the design or inserting Trojans into them [3], [21]. In split manufacturing [21], the layout of the design is split into two: the front end of line (FEOL) layers and the back end of line (BEOL) layers, which are then fabricated separately in different foundries, as illustrated in Fig. 3. The FEOL layers consist of transistors and lower metal layers (for example, metal 4 and lower layers [21]) and the BEOL layers consist of the top metal layers (for example, metal 5 and higher layers [21]). This corresponds to the partitioning of a gate level netlist into blocks where the transistors and wires inside a block form the FEOL layers, and the top metal wires connecting the blocks and the IO ports form the BEOL layers. Postfabrication, the FEOL and BEOL wafers are aligned, integrated, and tested [3], [21], [22].

The asymmetric nature of the metal layers facilitates split manufacturing. The top BEOL metal layers are thicker and have a larger pitch than the bottom FEOL metal layers. In one embodiment, the fabricated FEOL and BEOL wafers are integrated using electrical, mechanical, or optical alignment techniques and tested for defects by a system integrator [21]. In another embodiment, the FEOL wafer is fabricated in an advanced foundry and then sent to a trusted second foundry where the BEOL layout is built on top of it [21].

1) *Threat Model:* Split manufacturing may improve the security of an IC as the FEOL and BEOL layers are fabricated separately and combined postfabrication [3].

This prevents a single foundry (especially the FEOL foundry) from gaining full access to the IC. The attacker in the foundry that manufactures the FEOL wafer has the GDSII layout file of the design, and can reverse engineer it to obtain the gate-level netlist [5]. Thus, an attacker can always reverse engineer the FEOL parts of the circuits. However, he/she does not have information about the BEOL parts. Without the BEOL layers, however, the attacker can neither identify the “safe” places within a circuit to insert trojans nor pirate the designs. The attacker in the FEOL foundry gains knowledge about most of the design (the transistors and the lower metal layers) except for the missing BEOL connections. Once the attacker determines these missing BEOL connections, he/she can reconstruct the original design.

*Example 3:* Consider the 1985 International Symposium on Circuits and Systems (ISCAS-85) combinational logic benchmark circuit C17, shown in Fig. 4. It is partitioned into partition A (light colored) and partition B (dark colored). Typically, the wires within a partition (local wires except Vdd and clock) are assigned to lower metal layers. The wires that span the partitions and I/O ports are assigned to higher metal layers. This makes routing easier [24]. The nets connecting the input ports I1–I5 to the corresponding

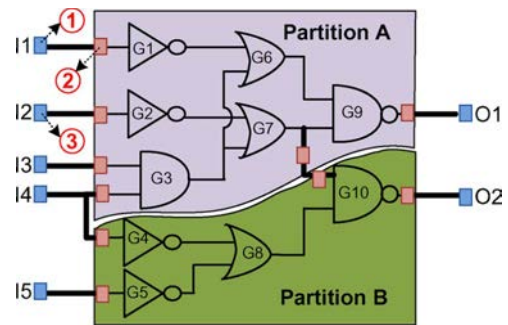


**Fig. 4. Secure processor design flow to thwart insider attacks. The design team and the foundries are not trusted. The trusted security and integration teams perform logic encryption, integration, and Trojan detection (shaded blocks) by unlocking the design.**

inputs of the gates G1–G5 use the BEOL layers. The nets connecting the output of gates G9 and G10 to output ports O1 and O2, respectively, use the BEOL layers. The net that connects the output of G7 to one of the inputs of G10 also uses the BEOL layers. Without the BEOL connections, an attacker in the foundry does not know the connectivity among input ports, output ports, and partition pins.

2) *Previous Work*: In [25], an algorithm to select wires for the BEOL layers is provided. A formal notion of an attacker's inability to figure out the missing BEOL connections is provided. However, this approach has a significant performance overhead, potentially superseding the benefits of a high-end FEOL foundry. Furthermore, it overlooks the vulnerability of the design to proximity attack, when a design is synthesized by conventional physical design tools. One can also leverage the 3-D manufacturing technology where security-sensitive components can be placed in one layer and manufactured in a trusted low-end foundry, and other components of the design can be placed in another layer and manufactured in an untrusted high-end foundry [26]. Feasibility of split manufacturing for analog designs has been demonstrated [27].

*Criterion 3 (Resilience to Proximity Attack)*: The attacker in the foundry should not be able to determine the missing FEOL connections from the BEOL connections.

*Problem 3*: Naive split manufacturing is vulnerable to proximity attack. This attack exploits the heuristic that floorplanning and placement (F&P) tools use to reduce the wiring (delay) between the pins to be connected [24]—place the partitions close by and orient the partitions. This heuristic of most F&P tools is a security vulnerability that can be exploited by an attacker in the FEOL foundry who does not have access to the BEOL layers. Thus, the attacker simply makes the missing connections between the two closest compatible<sup>2</sup> pins.

*Example 4*: Consider the locations of partition pins and IO ports of the F&P C17 benchmark as shown in Table 1. Consider the input port  $P_{I1,IO,in}$ ,<sup>3</sup> which is connected to pin  $P_{I1,A,in}$  in partition A. The locations of  $P_{I1,IO,in}$  and  $P_{I1,A,in}$  are (0, 6) and (1, 6), respectively. The distance between these two pins is 1 unit. Now, consider another input port  $P_{I3,IO,in}$ . The distance between  $P_{I3,IO,in}$  and  $P_{I1,A,in}$  is 1.414 units. Thus, the closest possible pin to  $P_{I1,A,in}$  is  $P_{I1,IO,in}$ . Hence, an attacker will connect these two pins in the netlist for the corresponding missing BEOL connection. Similarly, he/she can connect all the other partition pins with their closest pins and reconstruct the original design. An attacker can

**Table 1** Proximity Attack on Split Manufacturing. X-Y Coordinates of the Pins in Partitions A and B and IO Ports of F&P C17 Design Shown in Fig. 4. The Coordinates Are Shown as Absolute Units for Ease of Understanding

| Partition A    |             | Partition B     |             | Input & Output  |             |
|----------------|-------------|-----------------|-------------|-----------------|-------------|
| Pin            | XY location | Pin             | XY location | Port            | XY location |
| $P_{I1,A,in}$  | (1,6)       | $P_{I4,B,in}$   | (1,2)       | $P_{I1,IO,in}$  | (0,6)       |
| $P_{I2,A,in}$  | (9,5)       | $P_{I5,B,in}$   | (1,0)       | $P_{I2,IO,in}$  | (10,6)      |
| $P_{I3,A,in}$  | (1,5)       | $P_{G7,B,in}$   | (7,2)       | $P_{I3,IO,in}$  | (0,5)       |
| $P_{I4,A,in}$  | (1,4)       | $P_{G10,B,out}$ | (7,0)       | $P_{I4,IO,in}$  | (0,4)       |
| $P_{G9,A,out}$ | (9,4)       |                 |             | $P_{I5,IO,in}$  | (0,0)       |
|                |             |                 |             | $P_{O1,IO,out}$ | (10,5)      |
|                |             |                 |             | $P_{O2,IO,out}$ | (8,0)       |

perform a proximity attack to recover most of the missing BEOL connections [23]. Thus, naive split manufacturing is vulnerable to the proximity attack.

### C. DfTr 3: IC Camouflaging

Camouflaging is a layout-level technique that hampers an attacker from extracting a gate-level netlist of a circuit from the layout through imaging different layers. In one embodiment of IC camouflaging, dummy contacts are used [28]. Contacts are conducting materials that connect two adjacent metal layers or a metal layer 1 and poly. They pass through the dielectric that separates the two connecting layers. While a conventional contact (true contact) has no gap, a dummy contact has a gap in the middle and fakes a connection between the layers, as shown in Fig. 5. Such dummy contacts are used to design standard cells that look alike irrespective of their functionality. For example, NAND and NOR standard cells can be designed to look alike using dummy contacts. When deceived into incorrectly interpreting the functionality of the camouflage gate, the attacker may extract a netlist that is different from the original netlist.

1) *Threat Model*: To prevent reverse engineering, a designer camouflages certain gates in the design.<sup>4</sup> For example, the OR gate G7 in Fig. 6 is camouflaged. This design with camouflaged gates is then manufactured at a foundry. The manufactured IC is sold in the market. An attacker can reverse engineer an IC by depackaging, delayering, imaging the layers, and extracting the netlist. However, in the extracted netlist, the functionality of the camouflaged gates is unknown. For example, in Fig. 5, the functionality of G7 is unknown and an attacker assigns an arbitrary two-input function to it. Consequently, an attacker may obtain an incorrect netlist. Additionally, if the target IC is sold in the market, an attacker can buy one of them. In this way, an attacker can gain access to good I/O pairs.

2) *Previous Work on IC Camouflaging*: An IC camouflaging technique can leverage unused spaces in an IC and fill

<sup>2</sup>Two pins are compatible if one pin is the output of a gate or an input port, and the other pin is an input of a gate or an output port.

<sup>3</sup> $P_{Net,Partition,Direction}$  denotes a partition pin or an IO port. Net is the name of the wire in the original design. Partition represents either the partitions A or B or the IO port. Direction of a pin can be either in or out.

<sup>4</sup>A designer does not camouflage all the gates in the design because of the power, area, and delay overhead of the camouflaged gates.

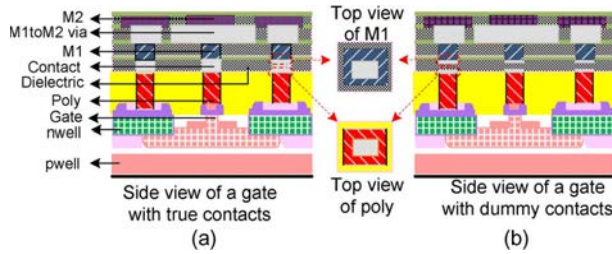


**Fig. 5. Dummy contact-based IC camouflaging. (a) A logic gate with true contacts. (b) A logic gate with dummy contacts. The top view, which is used by an attacker in reverse engineering, is identical for both true and dummy contacts [28].**

them with standard cells [29]. The outputs of these filler cells will not drive any active logic. Hence, an attacker can identify and discard them while extracting the netlist. One can also camouflage a design by using programmable standard cells [30]. Postfabrication, these cells will be programmed using a control input. However, such control inputs have to be stored on-chip which requires a tamper-proof nonvolatile memory. Similar to introducing dummy contacts, a designer can also create dummy channels, which will result in nonfunctional transistors [31] and can be used to deceive an attacker.

3) *Criteria for IC Camouflaging*: An IC camouflaging technique should satisfy two criteria: a) incorrect outputs should be produced on attempts to try the incorrect one of many possible functionalities of a camouflaged gate; and b) an attacker should not be able to retrieve functionality of the camouflaged gates.

*Criterion 4 (Output Corruption)*: The objective of the defender (designer) is to prevent his/her IP from being copied by an attacker in the foundry and to prevent black-box usage. The attacker does not know the functionality of the camouflaged gates. Hence, he/she will try one of the many possible functionalities of each camouflaged gate and in turn expect the design to become functional (i.e., to produce correct outputs). In the worst case, he/she has to perform a brute force attack by trying out all possible

functionalities of all camouflaged gates. The objective of the defender is to force the attacker to perform a brute force attack. Therefore, the defender needs the camouflaged design to produce incorrect outputs on incorrect functionality assignment to camouflaged gates. A defender has to camouflage the design such that an attacker, with the knowledge of the publicly available IC camouflaging objectives and algorithms, is not able to obtain the correct outputs by trying an incorrect functionality. This can be done by minimizing the correlation between the corrupted and original outputs, and thus by maximizing the ambiguity for the attacker similar to logic encryption and split manufacturing. The optimal point is again where 50% of the outputs are corrupted upon trying an incorrect functionality.

*Problem 4*: Random selection of gates for IC camouflaging fails to ensure the production of incorrect outputs for an incorrect functional assignment to camouflaged gates.

*Example 5*: Consider the circuit shown in Fig. 7. The camouflaged gate C1 can implement one of {XOR, NAND, NOR}. This circuit produces incorrect outputs for four out of the 16 possible input patterns, because the output C1 does not corrupt the output O1 for most of the input patterns.

*Criterion 5 (Difficult-to-Break Camouflaging)*: A camouflaging technique should ensure that the functionalities of the camouflaged gates should not be resolvable, even when an attacker has access to good I/O pairs.

*Problem 5*: Random selection of gates for IC camouflaging fails to ensure that the functionalities of the camouflaged gates are hardly resolvable.

*Example 6*: Consider the camouflaged gate C1 in Fig. 8. The functionality of C1 can be resolved to be XOR by applying 010XXXX at the inputs. This input pattern will justify the inputs of C1 to 00 and sensitize the output of C1 to O1. If O1 is 0, then the functionality of C1 is resolved as XOR. Otherwise, the functionality of C1 can be resolved to

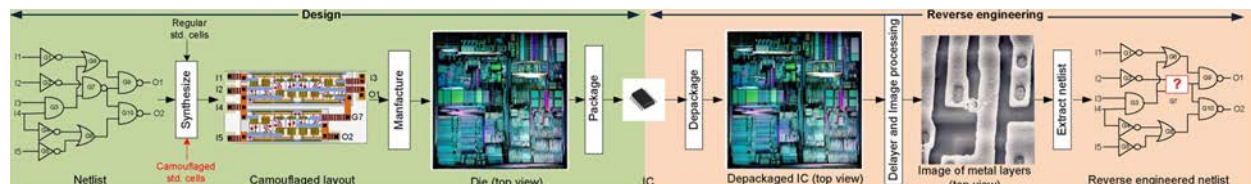


**Fig. 6. IC camouflaging in an IC design flow. A design is synthesized into layout by using both regular standard cells and camouflaged standard cells. This layout is manufactured to obtain the IC. An attacker buys the camouflaged IC and depackages it. He/she delays and images the metal layers and transistors. He/she then obtains the gate-level netlist by processing those images. However, the functionality of camouflaged cells cannot be resolved. For example, the functionality of G7 cannot be resolved [41].**

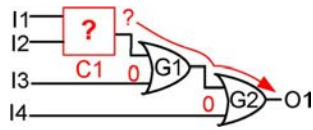


**Fig. 7. Problem with IC camouflaging.** Camouflaged gate C1 can be one of {XOR, NAND, NOR}. On incorrectly assigning the functionality of the camouflaged gate, wrong outputs are produced for only four out of 16 possible input patterns [41].

be NOR by applying 110XXXX at the inputs. This input pattern will justify the inputs of C1 to 10 and sensitize the output of C1 to O1. If O1 is 0, then the functionality of C1 is resolved as NOR. Otherwise, the functionality of C1 is resolved as NAND. Thus, a naive selection of gates for camouflaging may not offer security.

#### D. DfTr 4: Trojan Activation

Hardware Trojans inserted into a design at a foundry can be identified from their malicious behavior or by their additional power consumption [4], [32]–[34]. However, most of the Trojans remain dormant and they get activated only in rare conditions. Thus, to identify a Trojan one has to increase the switching activity within the Trojan circuit. This way a defender can activate the Trojan and observe its malicious behavior or can increase its power consumption, resulting in its detection [32].

1) *Threat Model*: As shown in Fig. 9, a designer synthesizes the design and generates the layout. He then characterizes its power and delay characteristics. An attacker in the foundry can insert Trojans into the design. The manufactured ICs with Trojans are then sent to the designer. The designer measures the power and delay characteristics of the manufactured IC by applying input patterns and

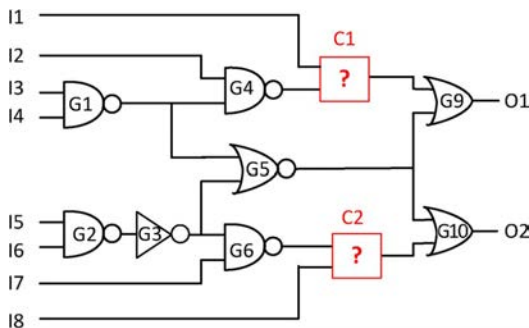


**Fig. 8. Problem with IC camouflaging.** C1 and C2 are camouflaged gates that could implement one of {XOR, NAND, or NOR}. An attacker can identify the functionality of C1 as XOR by applying 010XXXX at the inputs and observing a 0 at O1. If he/she observes a 1, 110XXXX can be applied from the inputs. If O1 is 0, the functionality of C1 is resolved as NOR. Otherwise, the functionality of C1 is resolved as NAND. Similarly, the functionality of C2 can be resolved [41].

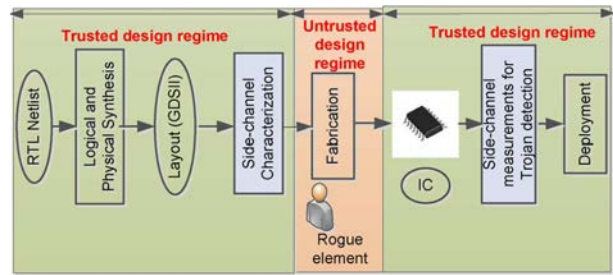


**Fig. 9. Threat model for hardware Trojans.** An attacker in the foundry can insert Trojans. A defender can characterize the side channel of the designs and compare it against the measurements made on the manufactured IC [4], [32]–[34].

compares them against the characterization. Any anomaly is considered as Trojan. The designer can use statistical and machine learning techniques to eliminate the effect of process variations [34], [35].

2) *Related Work*: Trojans are detected by characterizing the side channels such as power and delay on the golden model of the IC and comparing it against the measurements made on the manufactured IC. Based on the assumption that Trojans consume additional power, measurement of IC power dissipation can be used to detect Trojans [36]. Circuit delay characteristics have been used to detect Trojans [33], [35], [37]. While Jin and Makris [35] and Rajendran et al. [37] measure the delay of paths in the design and generate a delay fingerprint, Wang et al. [33] propose a method that measures the delays of all the paths using shadow latches. Potkonjak et al. [34] and Wei et al. [38] present nondestructive techniques to detect hardware Trojans in the presence of process variations where each IC component (e.g., a gate or an interconnect) has unique parameters. Unlike Salmani et al. [32] who use switching power, the techniques proposed in [34] and [38] use leakage power to detect Trojans. These techniques combine algebraic, numerical, and statistical methods with power and delay measurements to detect hardware Trojans. Furthermore, on-chip sensors are being incorporated to aid Trojan detection techniques [39].

*Criterion 6 (Sufficient Switching Activity by the Trojan)*: To detect Trojans by activation or by power side-channel analysis, a defender has to cause sufficient switching activity (for example,  $> 10^{-1}$ ) at the inputs of a gate to detect a Trojan.

*Problem 6*: One cannot always guarantee sufficient switching activity in all the gates. This is because certain gates have a low probability of transition.<sup>5</sup> Thus, in order to maintain the stealthy nature, an attacker can connect

<sup>5</sup>Probability of transition ( $P_{\text{Transition}}$ ) of a gate is the product of probability of obtaining a 1 and probability of obtaining a 0 at its output.

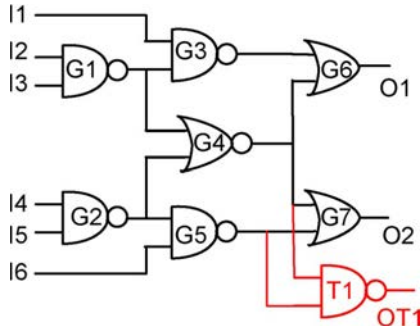


**Fig. 10. Problem with Trojan detection by activation. The inputs of a Trojan are connected to gates with low switching probability. Thus, one cannot cause sufficient switching activity within the Trojan to detect it.**

the inputs of a Trojan to the outputs of gates which have a low probability of transition.

*Example 7:* Consider the design shown in Fig. 10. T1 is the Trojan gate. The transition probability of the gates in the design is listed in Table 2. In this design, the inputs of the Trojan gate are connected to the output of the gates G4 and G5. This causes  $P_{\text{Transition}}$  to be 615/16384, which is less than the required value ( $> 10^{-1}$ ). In this way, the Trojan remains stealthy and can circumvent the Trojan detection technique.

#### IV. STRENGTHENING DfTr TECHNIQUES THROUGH VLSI TESTING PRINCIPLES

In this section, we present the application of VLSI test principles in the context of hardware security. First, we provide a background on a set of test principles that can be leveraged for hardware security. Then, we show how the DfTr techniques presented in Section III can be strengthened by leveraging these test principles.

**Table 2** Probability of Obtaining a 0 ( $P_0$ ), Probability of Obtaining a 1 ( $P_1$ ), Transition Probability ( $P_{\text{Transition}}$ ), and Average Number of Input Patterns Required to Obtain a Transition at the Output of the Gates Shown in Fig. 10. The Probabilities Are Calculated Based on the Assumption That Probabilities of Obtaining 0 and 1 ( $P_0, P_1$ ) at a Primary Input Are 0.5 and 0.5, Respectively [32]

| Gate | $P_0$            | $P_1$             | $P_{\text{Transition}}$ | Avg. # of input patterns |
|------|------------------|-------------------|-------------------------|--------------------------|
| G1   | $\frac{1}{4}$    | $\frac{3}{4}$     | $\frac{3}{16}$          | 4.3                      |
| G2   | $\frac{1}{4}$    | $\frac{3}{4}$     | $\frac{3}{16}$          | 4.3                      |
| G3   | $\frac{3}{8}$    | $\frac{5}{8}$     | $\frac{15}{64}$         | 3.3                      |
| G4   | $\frac{15}{16}$  | $\frac{1}{16}$    | $\frac{15}{256}$        | 16.0                     |
| G5   | $\frac{3}{8}$    | $\frac{5}{8}$     | $\frac{15}{64}$         | 3.3                      |
| G6   | $\frac{45}{128}$ | $\frac{83}{128}$  | $\frac{3735}{16384}$    | 3.4                      |
| G7   | $\frac{45}{128}$ | $\frac{83}{128}$  | $\frac{3735}{16384}$    | 3.4                      |
| T1   | $\frac{5}{128}$  | $\frac{123}{128}$ | $\frac{615}{16384}$     | 25.6                     |

As listed in Table 3, these principles can be utilized in order for the following to occur.

- Control the corruption at the outputs in three DfTr techniques: 1) in case of logic encryption, an incorrect key will result in incorrect outputs; 2) in case of split manufacturing, an incorrect BEOL connection will result in a design that produces incorrect outputs; and 3) in case of IC camouflaging, an incorrect assignment of a function to a camouflaged gate will result in incorrect outputs. The corresponding DfTr technique can then judiciously 1) insert the logic gates at proper locations; 2) swap block-level pins in split manufacturing; and 3) select the gates to be camouflaged. This requires the modeling of the injected corruption (corresponding to incorrect key, incorrect BEOL connections pins, or ambiguity of camouflaged gates) as faults, and making the DfTr decisions so as to favor the activation and propagation of these “faults.”
- Make sure that the DfTr technique is difficult to break. This enhancement applies to logic encryption, where encryption key needs to be protected, and to IC camouflaging, where the ambiguity regarding the one-of-many functionalities of camouflaged gates in the reverse-engineered netlist needs to be maintained.
- Convert low-activity regions to high-activity regions. In case of Trojan activation, the increased switching activity in the high-activity region exposes the Trojan.

#### A. Solution to Problem 1: Fault-Analysis Driven Logic Encryption

Logic encryption should be performed by inserting the key gates at carefully selected locations in the design so that 50% of the output bits are corrupted when an incorrect key is applied. The following observations relate logic encryption and fault analysis in IC testing and can be leveraged to guide the insertion of XOR/XNOR key gates for this purpose.

*Connection to Test Principle 1:* Application of a wrong key is analogous to excitation of a fault. For a wrong key, either a stuck-at-0 (s-a-0) or stuck-at-1 (s-a-1) fault will get excited, when XOR/XNOR gates are used for encryption. This is illustrated for the C17 circuit encrypted with one xor gate (E1) as shown in Fig. 11(b). If a wrong key ( $K1 = 1$ ) is applied to the circuit, the value of net B is the negated value of net A. This is the same as exciting an s-a-0 (when  $A = 1$ ) or s-a-1 (when  $A = 0$ ) fault at the output of G7, as shown in Fig. 11(a).

*Connection to Test Principle 3:* Corruption of an output due to a wrong key is analogous to the propagation of an excited fault to this output. This is illustrated for the circuit shown in Fig. 11(b). Let a wrong key ( $K1 = 1$ ) be

**Table 3** Enhancing DfTr Techniques Using Test Principles. The Number Within the Parentheses Indicates the Criterion/Test-Principle Number

| DfTr technique      | Criterion                          | Test principle(s)     | Effect                                                                                    |
|---------------------|------------------------------------|-----------------------|-------------------------------------------------------------------------------------------|
| Logic encryption    | Output corruption (2)              | Fault excitation (1)  | Application of a wrong key-bit                                                            |
|                     |                                    | Fault propagation (3) | Effect of a wrong key-bit propagating to an output                                        |
|                     |                                    | Fault masking (4)     | A pair of wrong key-bits masking each others effect                                       |
|                     | Difficult to break (3)             | Sensitization (2)     | Leaking of key-bits                                                                       |
| Split manufacturing | Resiliency to proximity attack (4) | Fault excitation (1)  | Wrong connection                                                                          |
|                     |                                    | Fault propagation (3) | Effect of a wrong connection propagating to an output                                     |
|                     |                                    | Fault masking (4)     | A pair of wrong connections masking each other's effect                                   |
| IC camouflaging     | Output corruption (5)              | Fault excitation (1)  | Assigning a wrong functionality to a camouflaged cell                                     |
|                     |                                    | Fault propagation (3) | Effect of a wrong function assignment propagating to an output                            |
|                     |                                    | Fault masking (4)     | A pair of camouflage cells with wrongly assigned functionality masking each others effect |
|                     | Difficult to break (3)             | Sensitization (2)     | Leaking of functionality of camouflage cells                                              |
| Trojan activation   | Increased switching activity (6)   | Controllability (5)   | Increased power consumption in Trojan circuits                                            |

applied to the circuit. For the input pattern 00000, an s-a-0 fault gets excited at the output of E1 and propagated to both outputs. The value at the output of the gate E1 is 0 instead of 1, and the output is 11 instead of the correct output 00. For the input pattern 01110, even though the s-a-0 fault gets excited at the output of E1, the output is 00, which is the same as the functional output, as the fault effects have been blocked.

**Connection to Test Principle 4:** Cancellation of the effect of multiple wrong key bits is analogous to the masking of multiple excited faults in a faulty design. Consider the encrypted circuit in Fig. 11(c). When key bits (K1, K2, and K3) are 000, the correct functional output is 00 for the input pattern 00000. However, if the key bits are 111 (wrong key), the effect introduced by the xor key gate E1 is masked by the xor key gates E2 and E3, resulting in the undesired correct output 00.

**Meeting Criterion 1:** Insert XOR/XNOR gates such that a wrong key will affect 50% of the outputs. In terms of fault simulation, this goal can be stated as finding a set of faults, which together will affect 50% of the outputs when excited. To insert an XOR/XNOR gate, we need to determine the location in the circuit where, if a fault occurs, it can

affect most of the outputs for most of the input patterns. To determine this location, we use the concept of fault impact defined by (1). From a set of test patterns, we compute the number of patterns that detect the s-a-0 fault ( $number\ of\ test\ patterns_{s-a-0}$ ) at the output of a gate Gx, and the cumulative number of output bits<sup>6</sup> that get affected by that s-a-0 fault ( $number\ of\ O/Ps_{s-a-0}$ ). Similarly, we compute  $number\ of\ test\ patterns_{s-a-1}$  and  $number\ of\ O/Ps_{s-a-1}$

**FaultImpact**

$$= (\# \text{ of test patterns}_{s-a-0} \times \# \text{ of } O/Ps_{s-a-0}) + (\# \text{ of test patterns}_{s-a-1} \times \# \text{ of } O/Ps_{s-a-1}). \quad (2)$$

Upon inserting an XOR/XNOR gate for encryption at the location with the highest fault impact, an invalid key will likely have the most impact on the outputs (i.e., the wrong outputs appear). Upon inserting a sufficient number of

<sup>6</sup>In sequential circuits, one should calculate the fault impact assuming that all the flip-flops are scan flip-flops and attacker has access to scan chains. Hence, each scan flip-flop is considered as a pseudo primary input and pseudo primary output.

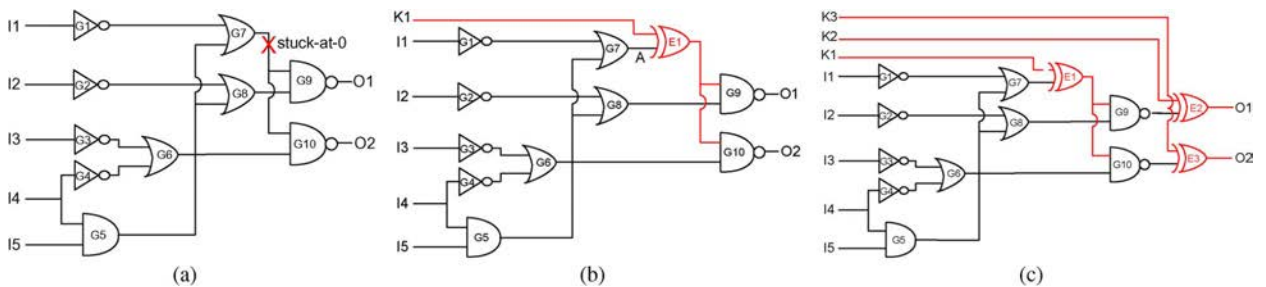


**Fig. 11.** Solution to problem 1 [20]: By relating logic encryption and fault analysis via testing principles—fault excitation, propagation, and masking—a designer can find optimal places in a design to insert the key gates to satisfy criterion 1. (a) A faulty circuit. (b) An encrypted circuit with a wrong key ( $K1 = 1$ ) equivalent to the faulty circuit. (c) A circuit encrypted with three xor gates (E1, E2, and E3).

XOR/XNOR gates, the 50% output corruption will be satisfied via a heuristic approach [18], [20].

### B. Solution to Problem 2: Difficult-to-Break Logic Encryption

To prevent attacks that aim at leaking the logic encryption key, key gates have to be inserted judiciously such that the attacker should be forced into decrypting the key bits using brute force. This way, even with full access to the netlist the attacker will not be able to circumvent the defense.

*Connection to Test Principle 2:* A key bit can be sensitized to an output only if the side input of every gate in the path to the output can be justified to a noncontrolling value. However, if a side input is not justifiable to a noncontrolling value due to an interference from another key bit, the target key bit cannot be sensitized to an output. Consequently, an attacker will not be able to decipher its value without knowing the value of the other key bit.

Consider the example circuit in Fig. 12 which is functionally identical to circuit in Fig. 2(b) but with the two key gates K1 and K2 inserted at different locations. The attacker cannot sensitize K1 to an output as the side input of G4 cannot be set to 0 (noncontrolling value of an OR gate). This is because an attacker neither knows the value of K2 nor can control K2. Similarly, K2 cannot be sensitized to an output due to K1. The attacker then has to decrypt these key bits together rather than individually.

*Meeting Criterion 2:* Logic encryption can be strengthened by inserting key gates with complex interferences among them. An interference exists between two key gates if the value of one key bit cannot be propagated or sensitized without controlling or knowing the value of the other key bit. By inserting the key gates such that they block each other's path, and/or they converge in some other gate, a difficult-to-break logic encryption that forces the attacker to perform brute force attempts can be implemented [13].

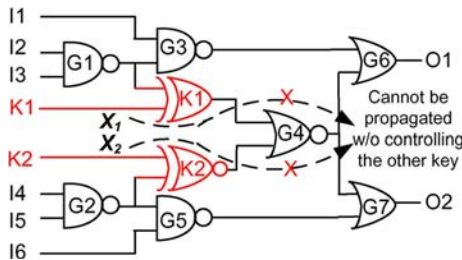


**Fig. 12. Solution to problem 2 [13]: An attacker cannot sensitize the effect of key bits K1 and K2 individually to the outputs. Hence, the attacker has to determine the values of K1 and K2 by brute force effort.**

This will prevent linear complexity attacks that target individual key gates and decipher individual key bits one at a time.

### C. Solution to Problem 3: Fault Analysis Driven Split Manufacturing

Proximity attack can be overcome by rearranging the pins such that they are no longer closest to the pins that they are supposed to connect. A proximity attacker will thereby be deceived into making the wrong BEOL connections. The objective of a designer then is to swap a sufficient number of pins such that the functionality of the deceiving netlist<sup>7</sup> differs from that of the original netlist. This objective can be quantified using the Hamming distance metric.

In the context of split manufacturing, the Hamming distance is defined as the number of output bits that differ on applying an input to the original design versus to the design with a pair of FEOL pins swapped. Ideally, 50% of the output bits should differ on applying any input for any swapped pin pair. This metric is formally defined as follows. For a circuit  $C$  which produces an output  $y$  for an input  $x$  on applying the valid key, split manufacturing of  $C$  should satisfy

$$\frac{\sum_{x \in X} \sum_{m \in M; m' \in M; m \neq m'} HD(y_{m,m',x}, y_{m',m,x})}{(2^{|M|} - 1) \times |y|} \times 100\% = 50\% \quad (3)$$

where  $X$  is the set of all inputs,  $M$  is the set of FEOL pins,  $y(m, m', x)$  is the output vector on applying the input  $x$  where the FEOL pins  $m$  and  $m'$  are connected correctly, and  $y(m', m, x)$  is the output vector on applying the input  $x$  where the FEOL pins  $m$  and  $m'$  are swapped.  $|x|$  and  $|y|$  denote the size of the input and output in bits, respectively.  $|M|$  is the number of FEOL pins.

To find a swapping pin for a target pin, similar to an attacker, the defender can build the list of candidate pins for that target pin. Then, he/she can randomly select the swapping pin from that list. Unfortunately, such random selections might not guarantee that the attacker will get a wrong output on making a wrong connection. Hence, one can use VLSI test principles to select the swapping pin for a target pin in order to achieve the 50% output corruption criterion, as illustrated in Fig. 13.

*Connection to Test Principle 1:* The fact that different values on the swapping and the target pin introduce an error is analogous to the excitation of a fault. Ideally, such pins must be swapped to introduce errors. Otherwise, the resulting design, even with the wrong connections, will still produce mostly correct outputs.

<sup>7</sup> A deceiving netlist is created by the defender by swapping the pins.

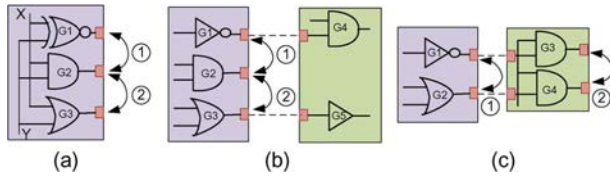


**Fig. 13.** Solution to problem 3 [23]: Applying IC testing principles to split manufacturing. (a) Pin that has a logical value opposite to that of the swapped pin is preferred (fault excitation). Values at  $P_{G1,A,out}$  and  $P_{G2,A,out}$  differ only when  $X = Y = 0$ ; Values at  $P_{G2,A,out}$  and  $P_{G3,A,out}$  differ in two cases:  $X = 1, Y = 0$  and  $X = 0, Y = 1$ . Thus,  $P_{G3,A,out}$  is selected as the swapping pin for  $P_{G2,A,out}$ . (b) If  $P_{G1,A,out}$  is selected as the swapping pin for  $P_{G2,A,out}$ , the wrong value will propagate only when the other input of G4 is 1. However, if  $P_{G3,A,out}$  is selected as the swapping pin, the buffer G5 will always propagate the wrong value (fault propagation). (c) The logical error introduced by swapping  $P_{G1,A,out}$  and  $P_{G2,A,out}$  is cancelled by swapping  $P_{G3,B,out}$  and  $P_{G4,B,out}$  (fault masking).

Connection to Test Principle 3: Corruption of an output due to two swapped pins is analogous to the propagation of an excited fault. Ideally, swapping should be done so that many outputs are corrupted (fault propagates to many outputs).

Connection to Test Principle 4: Cancellation of the effect of multiple swapped pin pairs is analogous to the masking of multiple excited faults. Sometimes, logical values corrupted by swapping pins in partition A can be restored to their original value because of swapping pins in partition B.

Meeting Criterion 3: Instead of randomly selecting the swapping pin, the pin that affects most of the outputs for most of the input patterns on swapping is selected. This accounts for fault activation, propagation, and masking scenarios. We define the fault impact metric, to select a swapping pin Y for a target pin X

$$\text{FaultImpact}_{X,Y} = \sum_{i=1}^{\text{\# of test patterns}} \text{\# of corrupted outputs.} \quad (4)$$

One can swap the target pin X with the swapping pin Y in the netlist and identify the cumulative sum of the corrupted output bits over a set of random test patterns. Fault impact quantifies the effect of swapping on the outputs of the design. Fault impact metric is used to select the swapping and target pins. The selected pins are then swapped and the netlist is updated. The above steps are repeated until all the partition pins and input ports are swapped or the Hamming distance value reaches 50%. For the ISCAS-85 benchmark designs, a designer achieves the 50% Hamming distance metric by swapping only a small set of

pins (< 20 for most designs) via a heuristic approach [23]. Apart from fault-analysis-based pin swapping, one can make proximity attack difficult by manufacturing FEOL layers to metal layer 1 and below in the untrusted FEOL foundry [40]. However, this will tremendously increase the cost of trusted BEOL foundry, as it has to support more metal layers.

#### D. Solution to Problem 4: Fault Analysis Driven IC Camouflaging

IC camouflaging should be performed by carefully selecting the logic gates to be camouflaged with the ultimate goal of meeting the output corruption criterion. The following observations relate IC camouflaging and fault analysis in IC testing and can be leveraged to guide the selection of gates to camouflage for this purpose.

Connection to Test Principle 1: Attempting the wrong functionality of a camouflaged gate is analogous to excitation of a fault. For the wrong functionality, either an s-a-0 or s-a-1 fault may get excited. This is illustrated in Fig. 14 for camouflaged gate C1 that could implement one of many functionalities of XOR, NAND, or NOR. The example pattern justifies the inputs of C1 to 01; if the actual functionality of C1 is XOR/NAND and the attempted (by the reverse engineer) functionality is NOR, an error is introduced. The same error is introduced when an s-a-0 at the output of C1 is excited. On the other hand, if the actual functionality is XOR and the attempted functionality is NAND, this pattern fails to introduce any corruption; a different pattern is needed in that case.

Connection to Test Principle 3: Corruption of an output due to attempting a wrong functionality of a camouflaged gate is analogous to the propagation of an excited fault. This is illustrated for the circuit shown in Fig. 7, where the corruption is propagated from the output of C1 to O1.

Meeting Criterion 4: Select gates to be camouflaged such that attempting wrong functionalities will affect 50% of the outputs. In terms of fault simulation, this goal can be stated as finding a set of faults, which together will affect

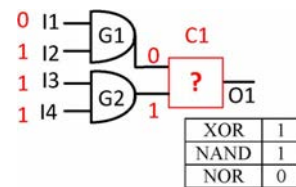


**Fig. 14.** Solution to problem 4 [41]: Camouflaged gate C1 can implement one of {XOR, NAND, or NOR}. Incorrect values are always propagated to the output.

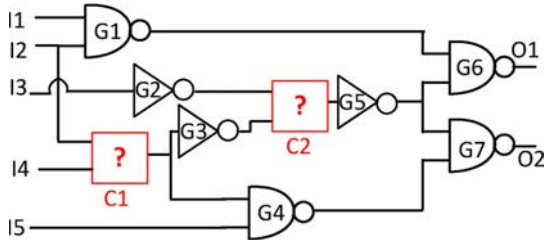


**Fig. 15. Solution to problem 5 [41]: An attacker cannot identify the functionalities of the camouflaged gates C1 and C2 without brute force effort.**

50% of the outputs when excited. For this purpose, we define the output corruption metric

$$\text{OutputCorruptibility} = \sum_{i=1}^{\text{\# of test patterns}} \# \text{ O/Ps,} \\ \text{with ambiguity activated and propagated.} \quad (5)$$

Camouflaging the gates with the highest output corruptibility values is expected to reduce the usefulness of the reverse engineered netlist [41].

#### E. Solution to Problem 5: Difficult-to-Break IC Camouflaging

Similar to implementing a difficult-to-break logic encryption, the attacker should be forced into using brute force in his/her attempts to identify the functionality of the camouflaged gates. Thus, the gates to be camouflaged should be selected judiciously; interference between the camouflaged gates increases the brute force effort for the attacker, forcing him/her to target camouflaged gates in large groups rather than individually.

*Connection to Test Principle 2:* An attacker can determine the functionality of a camouflaged gate by sensitizing its output to a primary output of the design. A gate's output can be sensitized to an output only if the side input of every gate in the path to the output can be justified to a noncontrolling value. However, if a side input is not justifiable to a noncontrolling value due to the ambiguity stemming from another camouflaged gate, the target gate's output cannot be sensitized to an output. Consequently, an attacker will not be able to resolve the functionality of the target camouflaged gate without knowing the functionality of the other camouflaged gate.

Consider the example circuit shown in Fig. 15 with two camouflaged gates C1 and C2. The attacker cannot sensitize C1's output to a primary output as the side input of G7 cannot be set to 1 (noncontrolling value of a NAND gate). This is because an attacker neither knows the functionality

of C2 nor can control C2. Similarly, C2's output cannot be sensitized to a primary output without knowing the functionality of C1. Thus, an attacker cannot identify the functionalities of C1 and C2 without brute force effort.

*Meeting Criterion 5:* IC camouflaging can be strengthened by creating complex interferences among the camouflaged gates. By selecting the camouflaged gates such that they block each other's path, and/or they converge in some other gate, a difficult-to-break IC camouflaging that forces the attacker into brute force can be implemented. This prevents linear complexity attacks that target individual camouflaged gates and identify individual gate functionalities one at a time [41], [42].

#### F. Solution to Problem 6: Increased Switching Activity

To increase the transition probability ( $P_{\text{Transition}}$ ) of a gate, one can introduce dSFFs [32] at its outputs. Through the dSFFs, one can easily control the input of the gate, increased  $P_{\text{Transition}}$ . This results in increased switching activity stemming from the Trojan circuit, causing the Trojan to consume more power. A defender can easily identify this additional power consumption during side-channel measurements and detect the Trojan. The dSFFs are used only when the designer needs to measure the side channels. During normal mode, they are bypassed and hence do not alter the functionality of the design. For this purpose, dSFFs are accompanied by a bypass gate (usually AND or OR).

*Connection to Test Principle 5:* Since the dSFFs are directly accessible to the designer through the scan chains, a designer can easily control them, injecting transitions. Hence, the  $P_{\text{Transition}}$  value of dSFF is 0.25, which is the same as that of the primary input. Adding the dSFF to a wire increases its  $P_{\text{Transition}}$  value as well as the that of the following wires. This results in increased switching activity in the design, thereby enabling Trojan detection.

Consider the circuit shown in Fig. 16. Here D1 is the dSFF gate and G8 is the bypassing gate. During normal

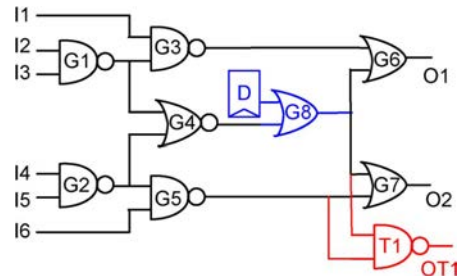


**Fig. 16. Solution to problem 6 [32]. Inserting dSFF (D1) increases the transition probabilities of G4 and G5. This results in increased switching activity, enabling Trojan detection. G8 is the bypassing gate.**

**Table 4** The  $P_{\text{Transition}}$  Values of the Gates in the Design Shown in Fig. 16 Before and After dSFF Insertion. Gates With Unchanged  $P_{\text{Transition}}$  Values Are not Listed

| Gate | Before dSFF insertion |                   |                         | After dSFF insertion |                   |                         |
|------|-----------------------|-------------------|-------------------------|----------------------|-------------------|-------------------------|
|      | $P_0$                 | $P_1$             | $P_{\text{Transition}}$ | $P_0$                | $P_1$             | $P_{\text{Transition}}$ |
| G8   | —                     | —                 | —                       | $\frac{15}{32}$      | $\frac{17}{32}$   | $\frac{255}{1024}$      |
| G6   | $\frac{45}{128}$      | $\frac{83}{128}$  | $\frac{3735}{16384}$    | $\frac{45}{256}$     | $\frac{211}{256}$ | $\frac{9495}{65536}$    |
| G7   | $\frac{45}{128}$      | $\frac{83}{128}$  | $\frac{3735}{16384}$    | $\frac{45}{128}$     | $\frac{83}{128}$  | $\frac{9495}{65536}$    |
| T1   | $\frac{5}{128}$       | $\frac{123}{128}$ | $\frac{615}{16384}$     | $\frac{85}{256}$     | $\frac{171}{128}$ | $\frac{14535}{65536}$   |

mode,  $D1$  is set to zero, retaining the original functionality of the circuit. When the designer wants to test for Trojans, he/she can set the desired input patterns at  $D1$ . Table 4 shows the  $P_{\text{Transition}}$  values of the gates before and after dSFF insertion. It can be seen that the  $P_{\text{Transition}}$  value of all the gates is greater than 0.1 after dSFF at their outputs.

**Meeting Criterion 6:** A designer can increase the  $P_{\text{Transition}}$  value of a gate by inserting dSFF at its output along with the bypassing gate. On inserting a sufficient number of dSFFs, one can guarantee that all the gates in the design will have  $P_{\text{Transition}}$  value greater than the desired value (for example, 0.1). An algorithm to insert dSFFs at optimal locations to have increased  $P_{\text{Transition}}$  with minimal power, area, and delay overhead is given in [32]. While dSFFs are used to increase the sensitivity of Trojans to switching power, one can also use them to deliver input patterns that maximize the sensitivity of Trojans to leakage power [43].

## V. DISCUSSION

### A. Computational Complexity of DfTr Techniques

DfTr techniques are computationally intensive because most of them use automatic test pattern generation (ATPG), which is an NP-complete problem [14]. However, efficient heuristics developed for practical circuits reduce this complexity to polynomial in the number of gates in the

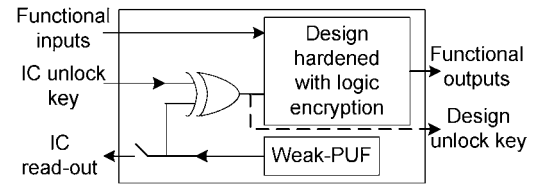


**Fig. 17.** Generating unique keys per chip using weak PUFs for logic encryption.

circuit [44]. Table 5 lists the complexity of DfTr techniques. In the case of logic encryption and camouflaging, the complexity is polynomial in the number of key gates or the number of gates to be camouflaged. However, the number of key gates or the number of gates to be camouflaged is relatively small (for instance, 128), making these techniques practical. In the case of proximity attack, the complexity is quadratic in the number of FEOL pins. Even though there may be thousands of FEOL pins, if not millions, it does not deter an attacker from applying the proximity attack. In the case of dSFFs, the complexity is linear in the number of gates in the design.

Furthermore, one does not need to apply logic encryption and camouflaging techniques to the entire chip. For example, in the case of processors, one can apply these techniques to only the controller units. Without the controller unit, an attacker cannot compute [45].

### B. Unique Unlock Keys Per Chip for Logic Encryption

In case of logic encryption, a designer should ensure that each chip has its own unlock key. Otherwise, a malicious user can use the key of one chip to unlock its pirated copies. In order to generate unique unlock keys per chip, one can leverage physical unclonable functions (PUFs). These are specialized circuits that leverage process variations to generate different outputs for the same set of inputs. One class of PUFs is called weak PUFs, which produce a unique signature per chip [46].

**Table 5** Computational Complexity of DfTr Techniques

| DfTr technique      | Criterion                      | Worst-case complexity                  | Complexity of efficient heuristics | Notation                                                                                                                        |
|---------------------|--------------------------------|----------------------------------------|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Logic encryption    | Output corruption              | $\mathcal{O}(2^{I \times K})$          | $\mathcal{O}(N \times K)$          | I - No. of inputs<br>K - No. of key-gates<br>N - No. of gates                                                                   |
|                     | Difficult to break             | $\mathcal{O}(2^{I \times K})$          | $\mathcal{O}(N^3 \times K^2)$      |                                                                                                                                 |
| Split manufacturing | Resiliency to proximity attack | $\mathcal{O}(2^{I \times M^2})$        | $\mathcal{O}(N \times M)$          | I - No. of inputs<br>M - No. of FEOL pins<br>N - No. of gates                                                                   |
| IC camouflaging     | Output corruption              | $\mathcal{O}(2^{I \times K \times M})$ | $\mathcal{O}(N \times K \times M)$ | I - No. of inputs<br>K - No. of camouflaged gates<br>M - No. of functions implemented by a camouflaged gate<br>N - No. of gates |
|                     | Difficult to break             | $\mathcal{O}(2^I \times K^M)$          | $\mathcal{O}(N^3 \times K^M)$      |                                                                                                                                 |
| Trojan activation   | Increased switching activity   | $\Omega(2^I)$                          | $\Omega(N)$                        | I - No. of inputs<br>N - No. of gates                                                                                           |

Logic encryption can be coupled with weak PUFs to generate unique unlock keys per chip, as shown in Fig. 17. A designer encrypts a design using logic encryption. The key to unlock this design is called “design unlock key.” The designer instantiates a weak PUF circuit in the IC along with the encrypted design, and sends the IC for fabrication. From the manufactured IC, the designer reads out the “chip key” produced by the weak PUF circuit. After this, the readout circuit is disabled. Because of the weak PUF circuit, each chip has its unique chip key. The designer XORs the chip-key with the design unlock key to compute the “chip unlock key.” The chip unlock key is given to the user who applies it to his/her chip to make it functional.

A malicious user cannot use his/her chip unlock key on a different chip, as that chip will have a different chip key because of the weak PUF circuit. In addition, the user cannot read out the chip key, as the readout circuit is disabled by the designer. Thus, by utilizing weak PUFs, a designer can produce unique unlock keys per chip.

### C. Which DfTr Techniques Should an IC Designer Use?

An IC designer can select any of the DfTr techniques depending on the trusted/untrusted entities in the IC design flow and the possible attacks that need to be thwarted (i.e., the threat model). An IC designer can use Table 6 as a guideline for selecting the appropriate DfTr technique(s).

Logic encryption and split manufacturing techniques provide an indirect protection against hardware trojans. Without the key or BEOL connections, a rogue element in the foundry will be unable to perform structural analysis to accurately identify safe places in the design to insert Trojans. Logic encryption protects the design IP against piracy and reverse engineering, and overbuilt ICs because, without the key, the chip will be nonfunctional, and thus, useless. Split manufacturing assumes the untrusted foundry but trusted end-user model; missing BEOL connections at the untrusted foundry will provide protection against IP piracy and IC overbuilding, but reverse engineering by end users will expose these missing connections.

IC camouflaging assumes the trusted foundry but untrusted end-user model; functionality of camouflaged gates, which is unknown to the end user, will provide protection against IP piracy and reverse engineering, but the dummy/real contact information that is available at the foundry may be used to circumvent the camouflaging technique, if the foundry is untrusted. Trojan activation only targets Trojans inserted at the foundry by increasing

**Table 6** DfTr Techniques, Their Threat Models, and Security Guarantees

| DfTr technique      | Defender | Trusted entity    | Attacker       | Attacks thwarted                 |
|---------------------|----------|-------------------|----------------|----------------------------------|
| Logic encryption    | Designer | –                 | Foundry        | Trojan insertion                 |
|                     |          |                   |                | IP Piracy                        |
|                     |          |                   |                | IC overproduction                |
|                     |          |                   | Malicious user | Reverse engineering<br>IP Piracy |
| Split manufacturing | Designer | BEOL foundry User | FEOL foundry   | Trojan insertion                 |
|                     |          |                   |                | IP piracy                        |
|                     |          |                   |                | IC overproduction                |
| IC camouflaging     | Designer | Foundry           | Malicious user | Reverse engineering<br>IP Piracy |
| Trojan activation   | Designer | User              | Foundry        | Trojan insertion                 |

the switching activity within the circuit. Designers have to choose a set of techniques that suit their business model. Using more than one defense techniques will provide layers of defense against a variety of attacks.

## VI. SUMMARY AND CONCLUSION

In this paper, we elaborated on four DfTr schemes: logic encryption, split manufacturing, IC camouflaging, and Trojan activation. We outlined various different threat models, ranging from untrusted foundry to untrusted end-user, as well as DfTr techniques as a defense against such threats. Designers have to choose a set of techniques that suits their business model. Though using more than one defense technique will provide layers of defense against a variety of attacks, a DfTr technique suitable for one threat model may not be suitable for another one.

We described not only the basic DfTr techniques, but also additional ways to further strengthen them via the use of a few basic VLSI test principles. By defining and quantifying security metrics and deciphering the relationship between the fundamentals of these DfTr techniques and VLSI testing principles, we show that these techniques can be enhanced through the use of VLSI testing tools. Furthermore, these techniques have to be reinforced with mathematical proofs.

In summary, this paper linked VLSI testing with hardware security, and thereby will 1) introduce hardware security issues to the test community and elicit how relevant the problems are to VLSI testing; 2) introduce the hardware security and trust community how VLSI testing tools [47], [48] and techniques can be leveraged for enforcing security; and 3) motivate computer-aided design tool developers to create dedicated tools to facilitate the manufacturing of trustworthy ICs. ■

## REFERENCES

- [1] International Technology Roadmap for Semiconductors (ITRS). [Online]. Available: <http://www.itrs.net/Links/2011ITRS/Home2011.htm>
- [2] DIGITIMES Research, “Trends in the global IC design service market.” [Online]. Available: <http://www.digitimes.com/news/a20120313RS400.html?chid=2>
- [3] Intelligence Advanced Research Projects Activity (IARPA), “Trusted Integrated Circuits Program,” 2011. [Online]. Available: <https://www.fbo.gov/utis/view?id=b8be3d2c5d5babbdfc6975c370247a6>
- [4] Defense Science Board (DSB), “High performance microchip supply,” 2005. [Online]. Available: <http://www.acq.osd.mil/dsb/reports/ADA435563.pdf>

- [5] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," in *Proc. IEEE/ACM Design Autom. Conf.*, 2011, pp. 333–338.
- [6] J. Roy, F. Koushanfar, and I. Markov, "EPIC: Ending piracy of integrated circuits," in *Proc. IEEE/ACM Design Autom. Test Eur.*, 2008, pp. 1069–1074.
- [7] R. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Design*, vol. 28, no. 10, pp. 1493–1502, Oct. 2009.
- [8] J. Rajendran, E. Gavas, J. Jimenez, V. Padman, and R. Karri, "Towards a comprehensive and systematic classification of hardware Trojans," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2010, pp. 1871–1874.
- [9] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 10–25, Jan.–Feb. 2010.
- [10] SEMI, "Innovation is at risk as semiconductor equipment and materials industry loses up to \$4 billion annually due to IP infringement," 2008. [Online]. Available: [www.semi.org/en/Press/P043775](http://www.semi.org/en/Press/P043775)
- [11] R. S. Chakraborty and S. Bhunia, "Security against hardware Trojan through a novel application of design obfuscation," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2009, pp. 113–116.
- [12] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *Proc. USENIX Security*, 2007, pp. 291–306.
- [13] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proc. IEEE/ACM Design Autom. Conf.*, 2012, pp. 83–89.
- [14] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. Boston, MA, USA: Kluwer, 2000.
- [15] Maxim Integrated, "DS28CN01: 1 kbit I<sup>2</sup>C/SMBus EEPROM with SHA-1 engine." [Online]. Available: <http://www.maximintegrated.com/datasheet/index.mvp/id/5369>
- [16] B. Barak et al., "On the (im)possibility of obfuscating programs," *J. ACM*, vol. 2, no. 6, pp. 1–48, 2012.
- [17] W. Griffin, A. Raghunathan, and K. Roy, "CLIP: Circuit level IC protection through direct injection of process variations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 5, pp. 791–803, May 2012.
- [18] J. Rajendran et al., "Fault analysis-based logic encryption," *IEEE Trans. Comput.*, 2013, DOI: 10.1109/TC.2013.193.
- [19] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 66–75, Jan./Feb. 2010.
- [20] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic encryption: A fault analysis perspective," in *Proc. IEEE/ACM Design Autom. Test Eur.*, 2012, pp. 953–958.
- [21] R. Jarvis and M. G. McIntyre, "Split manufacturing method for advanced semiconductor circuits," U.S. Patent 7 195 931, 2004.
- [22] M. Jagasivamani, P. Gadfort, M. Sika, M. Bajura, and M. Fritze, "Split fabrication obfuscation: Metrics and techniques," in *Proc. IEEE Symp. Hardware Oriented Security Trust*, 2014.
- [23] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Proc. IEEE/ACM Conf. Design Autom. Test Eur.*, 2013, pp. 1259–1264.
- [24] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*. New York, NY, USA: Springer-Verlag, 2002.
- [25] F. Imeson, A. Emtenan, S. Garg, and M. Tripunitara, "Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation," in *Proc. USENIX Security*, 2013, pp. 495–510.
- [26] J. Valamehr et al., "A 3-D split manufacturing approach to trustworthy system development," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 4, pp. 611–615, Apr. 2013.
- [27] K. Vaidyanathan et al., "Efficient and secure intellectual property (IP) design for split fabrication," in *Proc. IEEE Symp. Hardware Oriented Security Trust*, 2014.
- [28] J. P. Baukus, L. W. Chow, and W. Clark, "Integrated circuits protected against reverse engineering and method for fabricating the same using an apparent metal contact line terminating on field oxide," U.S. Patent 0 096 776, 2002.
- [29] J. P. Baukus, L. W. Chow, R. P. Cocchi, P. Ouyang, and B. J. Wang, "Camouflaging a standard cell based integrated circuit," U.S. Patent 8 151 235, 2012.
- [30] J. P. Baukus, L. W. Chow, R. P. Cocchi, P. Ouyang, and B. J. Wang, "Building block for a secure CMOS logic cell library," U.S. Patent 8 111 089, 2012.
- [31] J. P. Baukus, L.-W. Chow, J. W. M. Clark, and G. J. Harbison, "Conductive channel pseudo block process and circuit to inhibit reverse engineering," U.S. Patent 8 258 583, 2012.
- [32] H. Salmani, M. Tehranipoor, and J. Plusquellic, "New design strategy for improving hardware Trojan detection and reducing Trojan activation time," in *Proc. IEEE Int. Workshop Hardware-Oriented Security Trust*, Jul. 2009, pp. 66–73.
- [33] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *Proc. IEEE Int. Workshop Hardware-Oriented Security Trust*, Jun. 2008, pp. 15–19.
- [34] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, "Hardware Trojan horse detection using gate-level characterization," in *Proc. IEEE/ACM Design Autom. Conf.*, 2009, pp. 688–693.
- [35] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *Proc. IEEE Int. Workshop Hardware-Oriented Security Trust*, Jun. 2008, pp. 51–57.
- [36] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan Detection using IC fingerprinting," in *Proc. IEEE Symp. Security Privacy*, May 2007, pp. 296–310.
- [37] J. Rajendran, V. Jyothi, O. Sinanoglu, and R. Karri, "Design and analysis of ring oscillator based design-for-trust technique," in *Proc. IEEE VLSI Test Symp.*, 2011, pp. 105–110.
- [38] S. Wei, S. Meguerdichian, and M. Potkonjak, "Gate-level characterization: Foundations and hardware security applications," in *Proc. IEEE/ACM Design Autom. Conf.*, 2010, pp. 222–227.
- [39] X. Zhang and M. Tehranipoor, "RON: An on-chip ring oscillator network for hardware Trojan detection," in *Proc. IEEE/ACM design Autom. Test Eur. Conf. Exhibit.*, 2011, DOI: 10.1109/DATE.2011.5763260.
- [40] K. Vaidyanathan, B. P. Das, E. Sumbul, R. Liu, and L. Pileggi, "Building trusted ICs using split fabrication," in *Proc. IEEE Symp. Hardware Oriented Security Trust*, 2014.
- [41] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proc. ACM Conf. Comput. Commun. Security*, 2013, pp. 709–720.
- [42] J. Rajendran, O. Sinanoglu, and R. Karri, "VLSI testing based security metric for IC camouflaging," in *Proc. IEEE Int. Test Conf.*, 2013, DOI: 10.1109/TEST.2013.6651879.
- [43] D. Duarte, Y.-F. Tsai, N. Vijaykrishnan, and M. Irwin, "Evaluating run-time techniques for leakage power reduction," in *Proc. IEEE Int. Conf. VLSI Design*, 2002, pp. 31–38.
- [44] M. Prasad, P. Chong, and K. Keutzer, "Why is ATPG easy?" in *Proc. IEEE/ACM Design Autom. Conf.*, 1999, pp. 22–28.
- [45] J. Rajendran, A. Kanuparthi, M. Zahrn, S. Addepalli, G. Ormazabal, and R. Karri, "Securing processors against insider attacks: A circuit-microarchitecture co-design approach," *IEEE Design Test*, vol. 30, no. 2, pp. 35–44, Apr. 2013.
- [46] U. Ruhrmair, S. Devadas, and F. Koushanfar, "Security based on physical unclonability and disorder *Introduction to Hardware Security and Trust*. New York, NY, USA: Springer-Verlag, 2012, pp. 65–102.
- [47] H. Lee and D. S. Ha, "HOPE: An efficient parallel fault simulator for synchronous sequential circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 9, pp. 1048–1058, Sep. 1996.
- [48] H. Lee and D. Ha, "An efficient forward fault simulation algorithm based on the parallel pattern single fault propagation," in *Proc. IEEE Int. Test Conf.*, 1991, pp. 946–955.

## ABOUT THE AUTHORS

**Jeyavijayan Rajendran** (Student Member, IEEE) received the M.S. degree from Polytechnic School of Engineering, New York University, Brooklyn, NY, USA, where he is currently working toward the Ph.D. degree at the Electrical and Computer Engineering Department.

His research interests include hardware security and emerging technologies.

Mr. Rajendran is the organizer of the annual Embedded Systems Security Challenge, a hardware security competition. His publications won the Best Student Paper Awards at the 2013 ACM Conference on Computer and Communications Security, the 2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, and the 2011 IEEE VLSI Design Conference. He won the third place in the 2013 Grande Finals of the ACM Student Research Competition; the ACM Student Research Competitions for Design Automation at the 2012 Design Automation Conference and the 2013 International Conference on Computer-Aided Design; the ACM PhD Forum for Design Automation at the 2014 Design Automation Conference; the Myron M. Rosenthal award at the 2011 NYU-Poly; the Service Recognition Award from Intel; and the third place at the 2011 Kaspersky American Cup. He is a student member of the Association for Computing Machinery (ACM).



**Ozgur Sinanoglu** (Member, IEEE) received the B.S. degrees in electrical and electronics engineering and in computer engineering from Boğaziçi University, Istanbul, Turkey, in 1999 and the M.S. and Ph.D. degrees in computer science and engineering from the University of California at San Diego, La Jolla, CA, USA, in 2001 and 2004, respectively.

He is an Associate Professor of Electrical and Computer Engineering at the New York University–Abu Dhabi, Abu Dhabi, United Arab Emirates. He has industry experience from Qualcomm as a senior DFT engineer post-Ph.D. His research interests include design for test, design for security, and design for trust for VLSI circuits, where he has around 120 conference and journal papers, and 15 issued and pending U.S. patents.

Prof. Sinanoglu is the recipient of the best paper awards at the 2011 VLSI Test Symposium and the 2013 ACM Conference on Computer and Communication Security. During his Ph.D., he won the IBM Ph.D. fellowship award twice.



**Ramesh Karri** (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of California at San Diego, La Jolla, CA, USA, in 1993.

He is a Professor of Electrical and Computer Engineering at the Polytechnic School of Engineering, New York University, Brooklyn, NY, USA. His research interests include trustworthy integrated circuits (ICs) and processors, high assurance nanoscale IC architectures and systems, very large scale integration (VLSI) design and test, and interaction between security and reliability. He has over 150 journal and conference publications in these areas. He is the Area Director for Cyber Security of the NY State Center for Advanced Telecommunications Technologies at NYU-Poly, Hardware Security Lead of the Center for Research in Interdisciplinary Studies in Security and Privacy (CRISSP, <http://crissp.poly.edu/>), cofounder of the Trust-Hub (<http://trust-hub.org/>), and organizer of the annual red team blue team event at NYU, the Embedded Systems Security Challenge (<http://esc.isis.poly.edu>).

Prof. Karri was the recipient of the Humboldt Fellowship and the National Science Foundation CAREER Award, and Best Student Paper Awards at the 2013 ACM Conference on Computer and Communications Security, the 2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, and the 2011 IEEE VLSI Design Conference. He cofounded and served as the Chair of the IEEE Computer Society Technical Committee on Nanoscale Architectures. He is a cofounder and steering committee member of the IEEE/ACM Symposium on Nanoscale Architectures (NANOARCH), Program Chair (2012) and General Chair (2013) of the IEEE Symposium on Hardware Oriented Security and Trust (HOST), Program Co-Chair (2012) and General Co-Chair (2013) of the IEEE Symposium on Defect and Fault Tolerant VLSI and Nanotechnology Systems, and the General Chair of the 2013 NANOARCH. He serves on several program committees, including the VLSI Test Symposium, the Design Automation Conference, the Hardware-Oriented Security and Trust Symposium, and the International Conference on Computer Design. He is the Associate Editor of the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN, and the *ACM Journal of Emerging Technologies in Computing*. He is an IEEE Computer Society Distinguished Visitor and has organized invited tutorials on trustworthy hardware (including at the 2012 VLSI Test Symposium, the 2012 International Conference on Computer Design, the 2013 IEEE North Atlantic Test Workshop, the 2013 Design Automation and Test in Europe, the 2013 International Test Conference, and the 2014 IEEE/ACM Design Automation Conference).



# A Primer on Hardware Security: Models, Methods, and Metrics

*The paper is a primer on hardware security threat models, metrics, and remedies.*

By MASOUD ROSTAMI, FARINAZ KOUSHANFAR, AND RAMESH KARRI

**ABSTRACT** | The multinational, distributed, and multistep nature of integrated circuit (IC) production supply chain has introduced hardware-based vulnerabilities. Existing literature in hardware security assumes *ad hoc* threat models, defenses, and metrics for evaluation, making it difficult to analyze and compare alternate solutions. This paper systematizes the current knowledge in this emerging field, including a classification of threat models, state-of-the-art defenses, and evaluation metrics for important hardware-based attacks.

**KEYWORDS** | Counterfeiting; hardware Trojans; IP piracy; reverse engineering; side-channel attacks

## I. INTRODUCTION

### A. Motivation

All algorithmically secure cryptographic primitives and protocols rely on a hardware root of trust to deliver the expected protections when implemented in software. Similarly, critical control and communication functions assume that the hardware platforms that they are implemented on are resilient to attacks. However, this is not the case as the following examples demonstrate.

Quo Vadis Labs has reported backdoors in an integrated circuit (IC) that is used in weapons control systems, nuclear power plants, and public transportation systems [1]. Reports from the U.S. Government indicate that counterfeit electronics are prevalent in computers, communications, automobile, control, and defense systems

[2], [3]. Ethical hackers showed that, by spoofing the communication signals between the parking payment smart card and the payment meter, one can add value onto these smart cards [4]. A demonstration at the 2012 Black Hat Conference showed a security vulnerability in hotel keycards [5]. The attacker exploited the small key search space offered by the cryptographic algorithm implemented in the keycard to expose the master key used to unlock all rooms. Microcontrollers are extensively used in embedded systems, and they are equipped with “fuse bits” to prevent unauthorized users from reading or modifying selected sections of its memory. A reverse engineer has been able to electrically reset these fuse bits and thereby gain modify/read access to the contents of its memory [6].

Cost, power consumption, performance, and reliability are considered while designing an IC. Security is an afterthought. An increase in the number and destructive power of hardware-based attacks has highlighted the need for securing the hardware root of trust side by side of power, cost, performance, and reliability optimizations. An emerging body of research in hardware security is addressing these problems [7]–[11]. While the progress in this field has been significant, the approaches taken by researchers has been largely *ad hoc*. Different assumptions are typically made concerning the hardware-based vulnerabilities, threats that exploit them, models for the considered threats, and defenses. Consequently, developed defenses cannot be compared against each other, even when they address the same hardware security problem.

Within this context, this paper systematizes the knowledge for a number of important contemporary problems in hardware security. It classifies hardware-based threats, defenses, and metrics to evaluate the effectiveness of the developed defenses.

### B. IC Supply Chain

We start by describing the IC supply chain shown in Fig. 1. This supply chain is distributed worldwide [7], [12],

Manuscript received April 15, 2014; revised June 30, 2014; accepted June 30, 2014.  
Date of current version July 18, 2014.

This work was supported in part by the U.S. Office of Naval Research (ONR) under Grant R17460 and the National Science Foundation (NSF) under Grants to Rice University (CNS-1059416) and NYU (CNS-1059328 and CCF-1319841), NYU/NYU-AD Center for Research in Information Security Studies and Privacy (CRISSP), and ARO W911NF-13-1-0272).

**M. Rostami** and **F. Koushanfar** are with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA (e-mail: masoud@rice.edu; farinaz@rice.edu).

**R. Karri** is with the Department of Electrical and Computer Engineering, Polytechnic Institute of New York University, Brooklyn, NY 11201 USA (e-mail: rkarri@nyu.edu).

Digital Object Identifier: 10.1109/JPROC.2014.2335155

0018-9219 © 2014 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See [http://www.ieee.org/publications\\_standards/publications/rights/index.html](http://www.ieee.org/publications_standards/publications/rights/index.html) for more information.

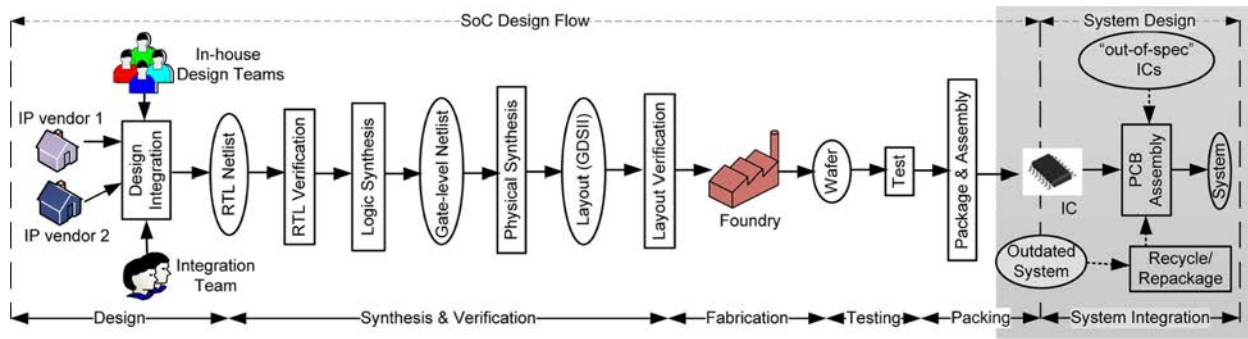


**Fig. 1. Semiconductor supply chain: IC design flow (only the steps and entities that are relevant to this paper are shown). System design (the dotted lines represent how the fake and low-quality components enter the supply chain). Source: [13].**

and the emerging hardware security problems arise because of the global trends in IC design, manufacturing, and distribution in this supply chain. Designing an IC involves procuring intellectual property (IP) designs from third-party design houses, designing some components inhouse, combining both, and generating the IC layout. A blueprint of the design (e.g., in terms of GDS-II layout format) is then sent to the foundry that develops a costly mask and manufactures the ICs. The ICs are then tested at the manufacturing site and often also at third-party test facilities. Finally, fault-free ICs are packaged and sold. There are multiple points within this supply chain where things can go wrong. The following hardware-based threats are possible.

- **Hardware Trojans:** An attacker either in the design house or in the foundry may add malicious circuits or modify existing circuits.
- **IP piracy and IC overbuilding:** An IP user or a rogue foundry may illegally pirate the IP without the knowledge and consent of the designer. A malicious foundry may build more than the required number of ICs and sell the excess ICs in the gray market.
- **Reverse engineering (RE):** An attacker can reverse engineer the IC/IP design to his/her desired abstraction level. He can then reuse the recovered IP or improve it.
- **Side-channel analysis:** An attacker can extract the secret information by exploiting a physical modality (power consumption, timing, or electromagnetic emission) of the hardware that executes the target application.
- **Counterfeiting:** An attacker illegally forges or imitates the original component/design.

### C. Systematization of Hardware Security Knowledge

Fig. 2 systematizes the hardware security knowledge centered around the attack method. The left column shows

the goals of the attack, and the right column shows the location of the attacker within the IC supply chain.

Fig. 3 presents this hardware security knowledge in terms of the hardware-based attacks, countermeasures, and metrics for evaluation. The left, attack column abstracts the scenarios pertaining to each attack class, the middle column summarizes the countermeasures, and right column shows the metrics for evaluation of the countermeasures. The description of the attack scenarios is application dependent.

A preliminary version of this paper has appeared in [13]. The paper is organized from the perspective of the state of the art in hardware-based attacks. Section II focuses on hardware Trojans. Section III details IP piracy and IC overbuilding. Section IV discusses reverse engineering. Section V explains side channels, and Section VI describes counterfeiting. For each attack, the threat model, the state-of-the-art defenses, and the metrics used to evaluate the defenses are systematized. Section VII concludes the paper.

## II. HARDWARE TROJANS

A hardware Trojan is a malicious modification to a circuit. The Trojan may control, modify, disable, or monitor the contents and communications of the underlying computing device [14]–[16]. Trojan detection is difficult for multiple reasons. First, the inherent opaqueness of the IC internals hinders detection of the modified components; conventional parametric IC testing methods have a limited effectiveness because of the classic observability issues, and destructive tests and IC RE are slow and expensive. Second, technology scaling to the limits of the device physics and mask imprecisions cause a nondeterminism in a chip's characteristics making the distinction between the process variation and Trojans hard. Finally, there is a large (uncharacterized) space in the IC for the possible Trojans.

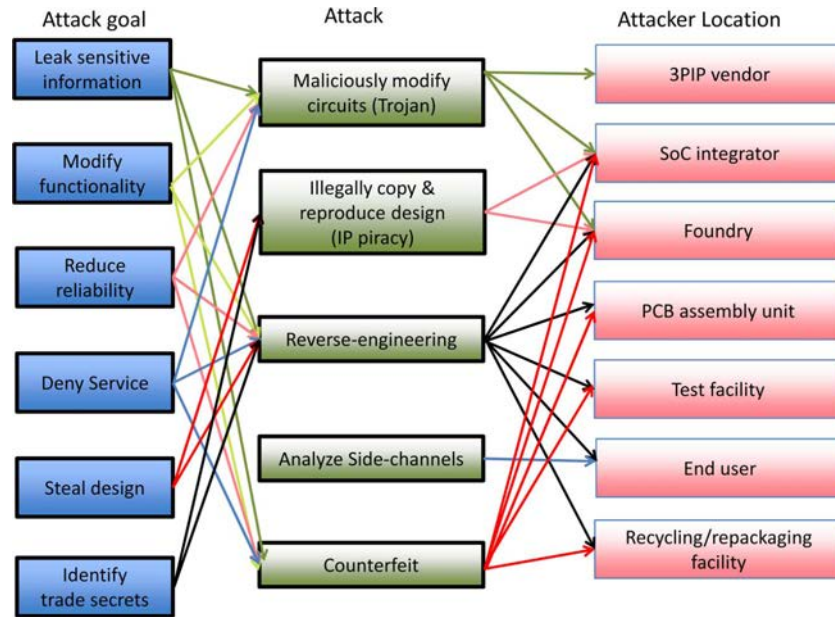


**Fig. 2.** Systematization of hardware security around the attack method. The left column shows the goals of the attack, and the right column shows the location of the attacker.

### A. Threat Models

Table 1 illustrates two common scenarios for a hardware Trojan attack. In the first scenario, an attacker in the foundry inserts a Trojan into the design by manipulating

the lithographic masks. These Trojans are in the form of addition, deletion or modification of gates [15], [16].

In the second scenario, a malicious IP is designed either by a rogue in the third-party IP (3PIP) design house

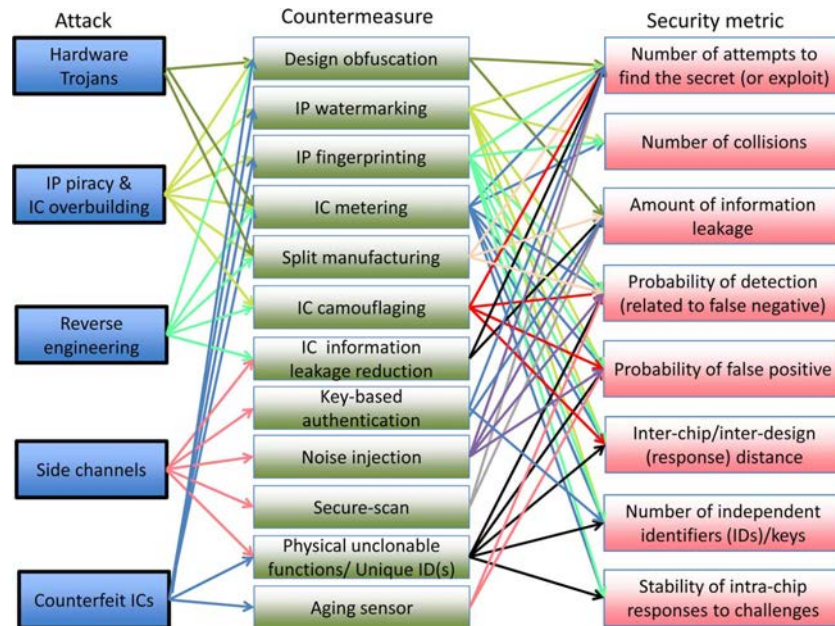


**Fig. 3.** Hardware security knowledge in terms of the hardware-based attacks, countermeasures, and metrics for evaluation. The left column abstracts the attack scenarios, the middle column abstracts the countermeasures, and the right column shows the metrics for evaluation.

**Table 1** Two Hardware Trojan Attack Scenarios: (i) by an Attacker in the Foundry; (ii) by a Rogue in the 3PIP Vendor. The Bullet (●) Depicts an Attacker, the Star (★) Represents a Defender, and the Dash (—) Indicates an Untrustworthy Entity

| Scenario | 3PIP Vendor | SoC Integrator | Foundry | User |
|----------|-------------|----------------|---------|------|
| 1        | —           | ★              | ●       | —    |
| 2        | ●           | ★              | —       | —    |

[15]–[17] or by a rogue in the inhouse design team [18]–[20]. It is unlikely that the malicious insider provides information about the inserted Trojan; without this information, the validation team may not be able to detect it.

### B. State-of-the-Art Defenses

Most techniques attempt to detect Trojans inserted in the foundry [15], [16]. There are at least two possible ways to detect this class of Trojans: invasive and non-invasive. Invasive (and semi-invasive) detection methods make the tested components unusable afterwards. These methods require costly, precision measurement equipments that only big silicon companies can afford [15], [16].

Noninvasive detection methods rely on external parametric and functional IC testing. These methods excite the circuit under test (CUT) with input patterns and measure the corresponding output values as well as the side channels, e.g., delay, quiescent leakage, and dynamic leakage [14]. Variants of functional and statistical tests are also used. Examples include transient power analysis [21], [22], path-delay measurements [23], gate-level characterization [24], [25], thermal profiling [26], or combinations of them [14], [26]. All of these techniques assume the availability of the full details of the circuit design, in addition to the statistical distribution of gate characteristics. The expected value of the characteristics of the IC is used as a reference model for detecting Trojans.

Several techniques combining invasive and noninvasive detection techniques have also been proposed. They attempt to model the structure of the IC by invasively testing a few, and then use the models in combination with noninvasive tests to detect Trojans [21].

Defenses against malicious 3PIP and insider attacks include self-monitoring [20] and static verification [19]. Trojans can also be prevented from activation by breaking the sequence/timing of events and by scrambling inputs supplied to the 3PIPs [18]. The integrator and the 3PIP vendor can also agree on a set of security properties which the integrator can verify [17].

### C. Metrics

1) Probability of detection: It is defined as the ratio of the number of Trojans detected by the technique to the total number of Trojans in the design. This metric equals to one minus the false positive rate [15], [16]. 2) Probability of false alarm: It is defined as the ratio of the number of

**Table 2** Scenarios for IP Piracy and IC Overbuilding. Obfuscation (O), Watermarking (W), Fingerprinting (F), and Metering (M) Are the Defenses

| Scenario | 3PIP Vendor | SoC Integrator | Foundry | User |
|----------|-------------|----------------|---------|------|
| 1        | O+W+F+M ★   | ●              | —       | —    |
| 2        | O+W+F+M ★   | —              | ●       | —    |
| 3        | —           | O+W+F+M ★      | ●       | —    |

Trojan-free designs that are incorrectly classified as Trojan to the number of Trojan-free designs [15], [16]. 3) The amount of time required to detect Trojans is important. In the case of Trojans inserted in the foundry, this time is often reported in terms of the number of applied test patterns. For 3PIP Trojans, this time is reported as the number of required clock cycles.

## III. IP PIRACY AND IC OVERBUILDING

An attacker with access to an IP or an IC can steal and claim ownership and/or can overbuild and sell them illegally [8], [27].

### A. Threat Models

Table 2 illustrates the threat. In scenario 1, the attacker in the integration house may pirate the 3PIP or use more than the licensed number of 3PIP instances. In scenario 2, the attacker in the foundry may pirate the 3PIP after extracting it from the layout of the design. In scenario 3, the attacker in the foundry may pirate the IC design and/or overbuild.

### B. State-of-the-Art Defenses

Five methods have been developed to thwart piracy and overbuilding: obfuscation, watermarking, fingerprinting, metering, and split manufacturing. In scenarios 1 and 2, the 3PIP vendor may protect his IP by obfuscating it, or by embedding his watermark, or by inserting a separate watermark in each instance of the IP (also called a fingerprint). In scenario 3, the integrator may obfuscate or embed his watermark or fingerprint the design before delivering it to the foundry.

1) *Watermarking*: A designer's signature is embedded into the design artifact [28]. The designer can later reveal the watermark and claim ownership of an IC/IP. Watermarks may include addition of black-hole states to the finite state machine (FSM) [29], addition of secret constraints during high level [30], logic and physical synthesis [31], and field-programmable gate array (FPGA) design [32].

Graph partitioning has found many applications in the IC design process: system design, behavioral synthesis, gate-level synthesis, physical design, packaging, and testing [33], [34]. One may encode the watermark as constraints during graph partitioning. For instance, one case

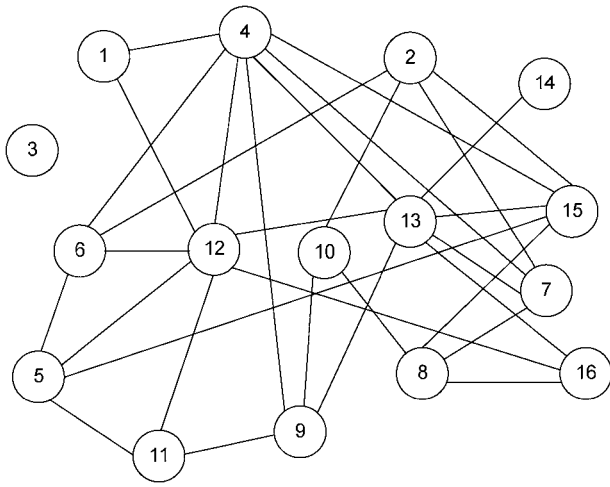


**Fig. 4. Motivational example for IP watermarking based on graph partitioning. Source: [33].**

constrains a set of nodes to be in the same partition. Alternately, a watermark can constrain the number of edges (edge cuts) spanning the partitions.

Consider embedding a watermark in the graph shown in Fig. 4. This graph has 16 nodes and 31 edges. One watermark may constrain pairs of vertices to be in the same partition. The number of possible watermarking solutions for different number of pairs and the quality of the corresponding solutions are depicted in Fig. 5. While there is only one solution for an edge-cut value of 9 (and hence this is not a good watermark constraint), there are 37 different solutions for an edge-cut value of 13. There is a delicate tradeoff between the number of possible solutions and the output quality that should be carefully considered.

A watermark should be: 1) unobtrusive, i.e., it should be oblivious to the functionality of the circuit; 2) robust,

i.e., it should be extremely difficult to remove; 3) unambiguous, i.e., it should yield conclusive proof of ownership; and 4) universal, i.e., it should be applicable to all designs [30], [35].

2) *Fingerprinting*: It helps the defender to track the source of piracy by embedding the signature of the buyer (for instance, his public key) along with the watermark of the designer [36]. When challenged, the designer can reveal the watermark to claim the ownership and the buyer's signature to reveal the source of piracy. For example, the power, timing, or thermal fingerprint of an IC is revealed on applying a set of input vectors.

Similar to watermarking, fingerprinting can also be applied during high-level, logic, and physical synthesis [36]. A technique that employs Kolmogorov–Smirnov statistical test for matching two probability distribution to identify whether a particular chip is fabricated at a particular foundry has been recently proposed in [37]. Another possibility is to use fingerprints derived from the static random access memory cells in the IC [38]. The recent Defense Advanced Research Projects Agency (DARPA) Supply Chain Hardware Integrity for Electronics Defense (SHIELD) program aims at uniquely (and irremovably) identifying chips, but the effectiveness of the approach is yet to be seen [39], [40].

A possible preventive measure of piracy is to register the authentic IC fingerprints using physical random functions (PUFs) and then match the dubious ICs against the PUF fingerprint database using efficient security protocols [41]. PUFs are physical functions that map the unique variations of IC's parameters to a digital output. This approach, which also works for legacy designs without any added overhead, was first proposed in [42]. Another possibility is to use the existing fingerprints from a chip's SRAM, e.g., [38]. A comprehensive survey of PUFs can be found in [43].

3) *Obfuscation*: Obfuscation hides the functionality and implementation of a design by inserting additional gates into it. In one type of obfuscation, XOR/XNOR gates [8], [44] and memory elements [45] are added. The obfuscated design will function correctly only on applying the correct value to these gates and memory elements.

In another type of obfuscation, the FSM of the design is obfuscated. An FSM can be obfuscated by adding extra states and/or transitions into it. Some states in the original FSM may be replicated [46], invalid transitions between states may be added [27], [47]–[49], unused states can be utilized [29], [50], [51], or additional states with no outward transitions, referred to as black hole states, can be added [29], [50], [51]. In all these techniques, only a valid key leads to the correct functionality; an invalid key leads the design into invalid states or transitions, and maybe into black hole states where the design will be stuck. Fig. 6 shows the obfuscated controller of an example FSM.

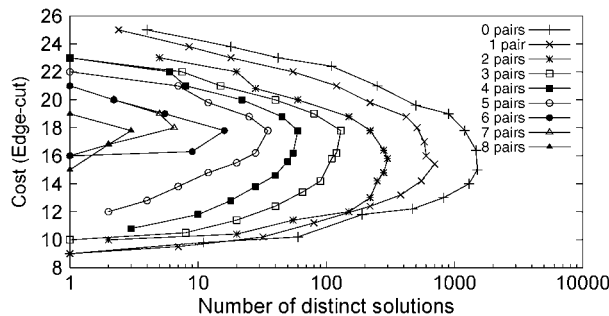


**Fig. 5. Watermarking: Number of possible watermarks versus quality of solutions for the graph when the following pairs of vertices are merged together: (16, 14), (6, 2), (16, 4), (9, 8), (5, 16), (9, 4), (11, 10), (9, 4). Source: [33].**

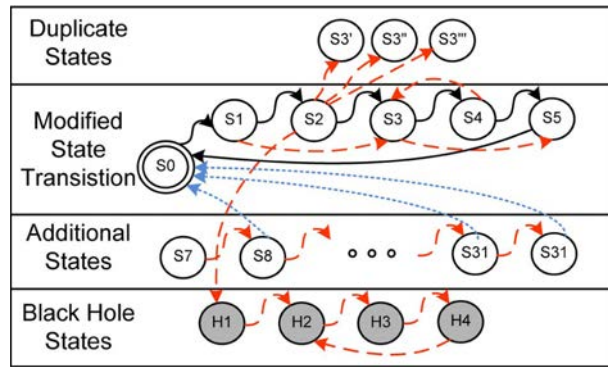


**Fig. 6. Obfuscating a controller. Approach 1: Existing states are replicated [46]. Approach 2: State transitions are modified [27], [47]–[49]. Approach 3: Additional states are added [29], [50], [51]. Approach 4: Black-hole states are added [29], [50], [51]. S0 through S6 are the states in the original FSM. All the other states are added for obfuscation. Solid edges are the state transitions in the original FSM. Dashed edges are state transitions from an invalid state to a valid state, on applying the valid key. Dotted edges are the state transitions from a valid state to an invalid state, on applying an invalid key or when key is withdrawn.**

4) *Metering*: It is a set of tools, methodologies, and protocols used to track a manufactured IC. In passive metering, part of an IC's functionality is used for metering [52]. The identified ICs are matched against their record in a database. This will reveal unregistered ICs or overbuilt ICs. In active metering, parts of the IC's functionality can be only accessed, locked, or unlocked by the designer and/or IP rights owners [29]. The difference between metering and obfuscation is that while metering uses a unique unlock key per IC, obfuscation just locks the IC.

5) *Split Manufacturing*: The layout of the design is split into the front-end-of-line (FEOL) layers and back-end-of-line (BEOL) layers. They are then fabricated separately in different foundries. The FEOL layers consist of transistors and other lower metal layers (say  $\leq M4$ ) and the BEOL layers consist of the top metal layers (say  $> M4$ ). Post-fabrication, the FEOL and BEOL wafers are aligned and integrated together using either electrical, mechanical, or optical alignment techniques. The final ICs are tested upon integration of the FEOL and BEOL layers [53], [54]. The asymmetric nature of the metal layers facilitates split manufacturing. The top BEOL metal layers are thicker and have a larger pitch than the bottom FEOL metal layers. Hence, a designer can integrate the BEOL and FEOL layers.

Split manufacturing is practical [55]. Ideally, an attacker should not be able to retrieve the missing BEOL connections by knowing the FEOL layers [56].

### C. Metrics

Metrics for watermarking include [30]: 1) collision, defined as the probability that a watermarking algorithm

generates the same solution for two different signatures; and 2) degradation in the quality of the solution. Ideally, degradation should have zero effect. For example, in the graph shown in Fig. 4, though merging more number of vertices reduces the probability of collision, it increases the edge cut, degrading the quality of the solution. Fingerprinting has the same metrics as for watermarking.

Metrics for obfuscation are: 1) the number of brute force attempts required to unlock the FSM or to determine the secret key [29], [44]; 2) the Hamming distance between the outputs of an obfuscated netlist on applying an incorrect key (or configuration) and the original netlist [45], [57]; 3) the number of input patterns that produce an incorrect output on applying an incorrect key to the design [27]; and 4) the strength of the generalized point function for provable obfuscation [51].

In addition to those used for obfuscation, metering uses the following metrics: 1) the average Hamming distance between the responses to the same challenge obtained from two different ICs; ideally, this value has to be 50%; 2) the average Hamming distance between the responses to the same challenge (or a repeatedly measured fingerprint) applied at different times and environmental conditions to the same IC; ideally, this value has to be 0%; 3) nondigital measures of distances  $1/2$ ; and 4) the number of independent IDs that can be generated.

Metrics for split manufacturing include: 1) the number of BEOL connections predicted by an attacker; and 2) the Hamming distance between the outputs of a netlist with BEOL connections predicted by an attacker and the original netlist.

## IV. REVERSE ENGINEERING

RE of an IC involves 1) identifying the device technology used in it [58]; 2) extracting its gate-level netlist [9]; and/or 3) inferring its functionality [59], [60]. Several techniques and tools have been developed to reverse engineer<sup>1</sup> ICs [61], [62]. RE can be misused to steal and/or pirate a design, identify the device technology, or illegally fabricate the target IC. The objective of the attacker is to successfully reverse engineer a design to a desired abstraction level. He can use the known input–output pairs to verify the functional correctness of the reverse-engineered design and/or to guide RE to extract the gate-level netlist of a competitor's IP and use it in one's own IC or illegally sell it as an IP.

The objective of the attacker is to successfully reverse engineer a design to its target abstraction level. The target level can vary depending on the objective of the attacker. If the objective is to pirate the design, the target abstraction level can be either the physical design level, the gate level, or the RT level. If the goal is to insert Trojans, the

<sup>1</sup>These tools enable RE to collect competitive intelligence, to verify a design, to check for commercial piracy, to determine patent infringements, and to detect hardware Trojans [59], [61], [62].

**Table 3** Threat Scenarios for RE. Obfuscation (O) and Camouflaging (C) Are the Defenses

| Scenario | 3PIP Vendor | SoC Integrator | Foundry | User |
|----------|-------------|----------------|---------|------|
| 1        | O *         | •              | —       | —    |
| 2        | O *         | —              | •       | —    |
| 3        | —           | O *            | •       | —    |
| 4        | O *         | —              | —       | •    |
| 5        | —           | O *            | —       | •    |
| 6        | —           | —              | C *     | •    |
| 7        | O *         | —              | C *     | •    |
| 8        | —           | O *            | C *     | •    |

target abstraction level can be either the gate level or the RT level.

### A. Threat Models

Table 3 illustrates the threat models for RE. In scenario 1, the attacker in the integration house can reverse engineer the 3PIP. The 3PIP vendor can protect his IP by obfuscating it. The foundry and the user are assumed to be untrustworthy. In scenario 2, the attacker in the foundry can extract the 3PIP from the layout of the IC. Similar to RE scenario 1, the vendor can obfuscate his IP before delivering it to the untrustworthy system-on-chip (SoC) integrator. In scenario 3, the attacker in the foundry can reverse engineer the IC. He can extract the transistor-level netlist from the layout [62], and then the gate-level netlist from it [63]. The integrator can protect the design by obfuscating it.

In scenarios 4–8, the user is the reverse engineer. He may depackage the IC, delayer it, image the layers, stitch those images, and extract the netlist. While a 3PIP vendor may obfuscate his IP (RE scenario 4), an integrator may obfuscate the layout (RE scenario 5). A trusted foundry might camouflage the layout (RE scenarios 6–8). This will provide an additional layer of defense beyond obfuscation (RE scenarios 7 and 8).

An algorithm to extract a gate-level netlist from transistors has been presented in [63]. Structural isomorphism can be used to extract the functionality of datapath modules [64]. Functionality of unknown units can be reverse engineered by performing behavioral matching against a library of components with known functionality such as adders, counters, register files, and subtractors [65]. The functionality of unknown modules can be identified by performing Boolean satisfiability analysis against a library of components with known functionality [66].

### B. State-of-the-Art Defenses

Obfuscation (see Section III-B) and camouflaging can thwart RE. In scenarios 1, 2, 4, and 7, a 3PIP vendor can obfuscate his IP. In scenarios 3, 5, and 6, an SoC integrator can obfuscate his design. A trusted foundry can camouflage the layout (scenarios 6–8) and add a layer of defense beyond obfuscation.

1) *Camouflaging*: This is a layout-level technique to hamper image-processing-based extraction of gate-level netlist. In one embodiment of camouflaging, the layouts of standard cells are designed to look alike, resulting in incorrect extraction of the netlist. The layout of NAND cell and the layout of NOR cell look different and hence their functionality can be extracted. However, the layout of a camouflaged NAND cell and the layout of camouflaged NOR cell can be made to look identical<sup>2</sup> and hence an attacker cannot unambiguously extract their functionality [67]–[70].

IC camouflaging can leverage unused spaces in a chip by filling them with filler cells [71]. One can camouflage a design by using programmable standard cells [69]. Post-fabrication, these cells may be programmed using a control input. One can also use dummy contacts—a dummy contact has a gap in the middle and fakes a connection between two metal layers—for camouflaging [67]. TSMC, a leading foundry, can manufacture dummy-contact-based camouflaging cells [67].

### C. Metrics

Metrics for RE include: 1) percentage of gates correctly extracted from a layout [9]; 2) percentage of gates whose functionality is correctly inferred [66]; and 3) the number of signals correctly matched between the signals in the component with known functionality and the signals in the target design [65].

Metrics for camouflaging include: 1) the number of brute force attempts required to identify the functionality of camouflaged gates [72], [73]; and 2) the Hamming distance between the outputs of the original netlist and the netlist in which the functionality of camouflaged gates is assigned by the attacker [72].

## V. SIDE-CHANNEL ATTACKS

Side-channel attacks exploit the leakage of secret information through a physical modality when an application is being executed on a system [10]. Side-channel attacks are powerful and have been able to break most existing important cryptographic algorithms [74].

Consider the RSA encryption algorithm which uses modular exponentiation with large exponents. An essential step in RSA encryption and decryption is computing  $m^e$ , where  $m$  is the message and  $e$  is either the public or private key. For an acceptable security level,  $m$  and  $e$  are required to be at least 1024-b numbers [75]. A naive approach to calculate  $m^e$  involves multiplying  $m$  by itself  $e - 1$  times. This approach requires  $e - 1$  multiplications, which is prohibitive.

To reduce the overhead, cryptographers use the square-and-multiply [75] algorithm. The pseudocode of the algorithm is shown in Algorithm 1.

<sup>2</sup>The contacts and vias are opaque to the RE tool as it processes the image templates for NAND and NOR standard cells.

---

**Algorithm 1:** Square and multiply algorithm for RSA exponentiation that calculates  $m^e$

---

```

1: Input:  $e = \sum_{i=0}^t x_i \times 2^i, b$ ;
2: Output = 1;
3: for  $i = t$  DOWNTO 0 do
4:   Output = Output2;
5:   if  $x_i = 1$  then
6:     Output = Output  $\times$   $m$ ;
7:   end if
8: end for

```

---

This algorithm first assigns 1 to the output. It then takes a pass through the bits of the exponent ( $e$ ) starting from the most significant bit. For each bit in  $e$ , the output is squared; if and only if the exponent bit is equal to “1,” a multiplication operation with the base value ( $b$ ) is performed. For example, while the naive approach for calculating  $x^{1026}$  takes 1025 multiplications, the square-and-multiply algorithm requires only 11 multiplications. Thus, for every logic 1 in the binary representation of the exponent, the square-and-multiply algorithm takes more cycles to finish the multiplication than the one in the naive approach. An adversary can guess the exponent by measuring the amount of time that a system takes to calculate an exponentiation (at the output). This type of adversarial key extraction relies on the execution delay analysis, and is called the timing side-channel attack [76].

Power consumption [10], electromagnetic (EM) emanations [77], photonic emissions [78], and acoustic noise of the system [79] are all correlated with the exponent, and can be used to extract the secret. Another side-channel attack against RSA exploits the Chinese remainder theorem (CRT) that is typically used to speed up its computation. If an adversary induces a fault during the CRT computation, the secret information can be obtained. Fault attacks can be launched using lasers, glitches in power supplies and clocks, and X-rays [80].

An attacker can scan out the secret key, when the key storing registers are connected as a scan chain<sup>3</sup> [81]–[83].

It has been shown that the power/timing consumption of PUF circuits is directly correlated with the process variation that PUF secrets are based upon. Therefore, PUFs are also shown to be susceptible to side-channel attacks [84]–[87].

### A. Threat Models

A realistic threat model must be developed first, and the defense should then vary depending upon the capabilities of the attacker in collecting the side-channel measurements. For example, securing a smart card is harder than securing the hardware of an offsite server against side-channel attacks; the adversary can manipulate

<sup>3</sup>A scan chain is a design-for-test structure that connects a set of flip-flops and makes them as a shift register. During test mode, scan chains are used to convert a sequential design to a combinational design.

the power and clock signals of a smart card, while he does not usually have access to power and clock systems of remote servers.

We recommend a variant of the threat model in [88]: Consider cryptographic functions of type  $F : K \times M \rightarrow D$ , where  $K$  is a finite set of keys,  $M$  is a finite set of messages, and  $D$  is an arbitrary set of ciphertext. The attacker is assumed to have no access to the values of  $k$  and  $F(k, m)$ , but he can measure/observe the characteristics of the physical implementation of  $F$ ,  $I_F$ . The objective of the side-channel attack is to find the value of the secret key(s). Even if the key(s) cannot be directly found, this attack reduces the search space for the key  $k$ .

This model assumes that the attacker has the full details of the implementation of  $I_F$  and can make one side-channel measurement per invocation of function  $F$ . If more than one observations are made for each invocation, it can be easily integrated within the model by adding variables to the output space. A side channel is a function  $f_{I_F} : K \times M \rightarrow O$ , where  $O$  is the set of possible observations, and  $f_{I_F}$  is known to the attacker.

### B. State-of-the-Art Defenses

1) *Leakage Reduction*: These techniques decrease the dependency between the side-channel traces of  $I_F$  and the secret information  $k$  [89]. For instance, consider the timing attack against RSA. The dependency between the timing information and the secret exponent can be reduced by performing “dummy” multiplication operations in Algorithm 1. This countermeasure incurs a 33% overhead and eliminates the leakage on timing channel.

The above countermeasure does not completely remove the threat of side-channel attacks. This is because other possible side-channel measurements (e.g., power consumption, EM, and acoustic noise of the computation unit) are still dependent on the secret multiplicands, although to a lesser extent. The dependence of side-channel information on system’s inputs is a systematic property of conventional complementary metal–oxide–semiconductor (CMOS) implementations. Several CMOS implementations have been proposed to mitigate this systematic leakage. For example, information leakage from power traces can be reduced by “smoothing” the power consumption using dynamic and differential logic [90], asynchronous logic [91], current-mode logic [92], or dual-rail with pre-charge logic [93]. The aforementioned circuit techniques cannot fully eliminate the side-channel leakage, because perfect symmetry in power and timing traces cannot be achieved due to inevitable process variations in CMOS process. However, these defenses can effectively reduce the signal-to-noise ratio (SNR) of the side channel.

2) *Noise Injection*: The SNR of the measurable side-channel information can be reduced by injecting artificial noise. This makes it more difficult for an attacker to

retrieve the secret key from the noisy side channel. In noise injection techniques, dummy circuits that consume random amount of power for each transaction, or by performing random operation independent of the secret information, are added [94] to the system.

The effects of added noise can be reduced by averaging over several samples or by applying advanced signal processing [95]. Therefore, noise injection does not provide a theoretical security but it does increase the required work of an attacker to extract the secret keys. It can be shown that decreasing the SNR of the side-channel information by a factor of  $K$  increases the number of required side-channel samples by a factor of  $K^2$  [96]. Temporal noise is an exception to this rule and increases the required samples by a factor of  $K$  [96].

3) *Key Update*: Frequently updating the secret key prevents the accumulation of side-channel information by the adversary [97]. This method uses a predefined sequence of keys (e.g., the output of a pseudorandom number generator) plus synchronized timings to ensure that the sequence of keys is consistent for both communicating parties. Several methods of key update and derivation, such as key tree [97], have been proposed.

If an estimate of the maximum information leakage rate per transaction ( $L_{MAX}$ ) were given, the keys should be updated before the amount of leaked information breaches a predefined level [95]. Unfortunately, to the best of our knowledge, there is still no reliable method of directly estimating  $L_{MAX}$  [98]. Instead, researchers explore new cryptographic primitives that have a theoretically proven bounded side-channel leakage per iteration [99]–[101].

4) *Side-Channel-Resistant PUFs*: Due to effectiveness of side-channel attacks against PUFs, it is imperative that circuit countermeasures, as proposed in [85], be used in future implementations. These countermeasures mitigate the correlation between the secret information and the measurable circuit delay/power consumption.

5) *Secure Scan Chains*: In a secure scan approach, mirror key registers are used in sensitive parts of the circuits [102]. These registers block unauthorized access to value of sensitive registers in the test mode of operation. In another approach, scan chains are divided into smaller subchains and access to them for regular users is randomized [82].

An adversary can combine the information leaked from several side channels to increase the effectiveness of the attack [74]. Likewise, several countermeasures, either in software or hardware, may be combined to increase the resiliency of the system against these attacks. In general, security experts advise against implementing security applications from scratch, and recommend leveraging open-source hardware or software implementations [103].

## C. Metrics

The two key metrics of side-channel attacks are: 1) the amount of secret information that is vulnerable; and 2) the number of samples from side channels needed to extract the secret information. Several metrics exist to quantify the amount of information leaked. An information-theoretic measure of side-channel leakage is quite desirable [88]. However, calculating a tight upper bound of information leakage may not be practical. Metrics have been proposed [98], [104], [105] to indirectly model and estimate the side-channel information leakage. Side-channel vulnerability factor (SVF) gauges the difficulty of finding the secret information from side channels [98]. SVF quantifies the correlation between the secret information and the patterns in the side-channel time trace. SVF is a system-level metric that can be applied to all types of side channels.

## VI. COUNTERFEITING

A counterfeit semiconductor component is an illegal forgery or imitation of the original component.<sup>4</sup> Counterfeiting is often performed by one of the many entities in the semiconductor supply chain, including new product vendors or secondary (recycled) IC vendors. In recent years, because of technological advances in 3-D packaging, fake ICs are hard to distinguish from the real ones.

Because of counterfeiting, the suppliers of the original components suffer loss. The poor performance of fake products, which are commonly lower quality or older generations of the original product, adversely impacts the overall system performance/reliability. It also harms the reputation of the authentic provider. Such fake products could potentially tamper the performance of weapons, airplanes, cars, or other critical applications that use them [11]. Although the common incentive for selling fake ICs is financial, the ease of inserting intentional hardware Trojans or spyware in fake ICs makes them a real security threat for the whole system which would eventually integrate the fake components.

### A. Threat Models

Table 4 illustrates the counterfeit IC threat models. In scenario 1, defective ICs, i.e., those which failed the manufacture-time testing and have been discarded, are used in consumer products [11]. An untrustworthy entity at the test facility can be the source of leaking defective ICs. In scenario 2, a dishonest entity in the IC supply chain mislabels a product and sells it as another IC potentially through a vendor [11]. Scenario 3 is similar to scenario 2 except for the following difference: While the designer

<sup>4</sup>Note that we make a distinction between the pirated/overbuilt ICs and fake ICs (although a clear distinction may be blurry in certain scenarios). IC piracy and overbuilding entail making ICs by illegally copying or stealing an authentic blueprint/IC during one of the design, synthesis, or production phases.

**Table 4** Threat Scenarios for Counterfeiting. The Shield Represents a Defender, and “—” Indicates an Untrustworthy Entity

| Scenario | Design | Test | Recycling/<br>Repackaging | PCB<br>Assembly |
|----------|--------|------|---------------------------|-----------------|
| 1        | —      | ●    | —                         | ★               |
| 2        | ★      | —    | ●                         | —               |
| 3        | —      | —    | ●                         | ★               |

employs proactive techniques to prevent counterfeiting in scenario 2, the assembly use reactive techniques to detect counterfeiting in scenario 3 [11].

## B. State-of-the-Art Defenses

1) *Hardware Metering and Auditing*: Hardware metering is a set of tools, methodologies, and protocols that enable postfabrication tracking of the manufactured ICs. Hardware metering may be passive, or active. In passive metering, part of the functionality of each IC can be specifically identified and used for metering, even for the ICs coming from the same mask [52]. The identified ICs may be matched against their record in a preformed database that could reveal unregistered ICs or overbuilt ICs (in case of collisions). In active metering, parts of the chip’s functionality can be only accessed, locked (disabled), or unlocked (enabled) by the designer and/or IP rights owners, using a high level knowledge of the design. Such knowledge is typically not accessible by the foundry or other supply chain entities [106].

2) *IC Fingerprints or PUFs*: See Section III.

3) *Device Aging Models/Sensors*: IC lifetime is influenced by a variety of phenomena [11], [42], [107], [108], such as negative temperature bias instability (NBTI), hot carrier injection, and electromagnetic migration. By employing sensors in ICs to measure these phenomena, an estimate of

chip lifetime can be found which would prevent counterfeiters from selling used chips as new ones. Measuring the previous usage of a device, while also detecting its authenticity, has been discussed quantitatively in [109].

4) *IP Watermarking*: See Section III.

## C. Metrics

The metrics for hardware metering, PUF fingerprinting, and watermarking are discussed in Section III. The two new metrics relevant to counterfeiting are: 1) probability of detection is the ratio of the number of counterfeit ICs detected by the technique to the total number of counterfeit ICs [110], [111]; and 2) probability of false positive is the ratio of the number of genuine ICs that are incorrectly classified as counterfeit ICs to the number of genuine ICs [42], [110].

## VII. CONCLUSION

In this paper, the threat models, state-of-the-art countermeasures, and metrics used to evaluate the defenses against hardware Trojans, IC and IP piracy, RE, side channels, and counterfeiting have been introduced. Until now, most evaluations of defenses have been informal and anecdotal. The authors believe that the metrics are an important first step in formalizing the evaluation of the strengths of defenses. Similarly, a consistent classification of threat models was not available. By organizing the threat/defense scenarios, we hope the countermeasures can be compared against each other based on the target threat model and the corresponding metrics. ■

## Acknowledgment

The authors would like to thank J. Rajendran (New York University, New York, NY, USA) for his input.

## REFERENCES

- [1] S. Skorobogatov, “Hardware assurance and its importance to national security,” 2012. [Online]. Available: <http://www.cl.cam.ac.uk/sps32/secnews.html>
- [2] U.S. Department of Commerce, “Defense industrial base assessment: Counterfeit electronics,” 2010.
- [3] 112th Congress, “Inquiry into counterfeit electronic parts in the department of defense supply chain,” Senate Report of the Committee on Armed Services, 2012.
- [4] J. Grand, J. Applebaum, and C. Tarnovsky, “Smart’ parking meter implementations, globalism, you aka meter maids eat their young,” 2009. [Online]. Available: [https://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-grand-appelbaum-tarnovsky-smart\\_parking.pdf](https://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-grand-appelbaum-tarnovsky-smart_parking.pdf)
- [5] “My Arduino can beat up your hotel room lock,” 2012. [Online]. Available: <http://demoseen.com/bhpaper.html>
- [6] A. Huang, “Hacking the PIC 18F1320,” 2007. [Online]. Available: [http://www.bunniestudios.com/blog/?page\\_id=40](http://www.bunniestudios.com/blog/?page_id=40)
- [7] Office of the Under Secretary of Defense For Acquisition, Technology, Logistics, “Defense Science Board (DSB) study on high performance microchip supply,” 2005. [Online]. Available: [www.acq.osd.mil/dsb/reports/ADA435563.pdf](http://www.acq.osd.mil/dsb/reports/ADA435563.pdf)
- [8] J. Roy, F. Koushanfar, and I. Markov, “EPIC: Ending piracy of integrated circuits,” *IEEE Computer*, vol. 43, no. 10, pp. 30–38, Oct. 2010.
- [9] R. Torrance and D. James, “The state-of-the-art in semiconductor reverse engineering,” in *Proc. IEEE/ACM Design Autom. Conf.*, 2011, pp. 333–338.
- [10] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” *Adv. Cryptol.*, pp. 388–397, 1999.
- [11] F. Koushanfar et al., “Can EDA combat the rise of electronic counterfeiting?” in *Proc. IEEE/ACM Design Autom. Conf.*, 2012, pp. 133–138.
- [12] SEMI, “Innovation is at risk as semiconductor equipment and materials industry loses up to \$4 billion annually due to IP infringement,” 2008. [Online]. Available: [www.semi.org/en/Press/P043775](http://www.semi.org/en/Press/P043775)
- [13] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, “Hardware security: Threat models and metrics,” in *Proc. Int. Conf. Comput.-Aided Design*, 2013, pp. 819–823.
- [14] F. Koushanfar and A. Mirhoseini, “A unified framework for multimodal submodular integrated circuits trojan detection,” *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 1, pp. 162–174, Mar. 2011.

- [15] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *IEEE Computer*, vol. 43, no. 10, pp. 39–46, Oct. 2010.
- [16] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 10–25, Jan./Feb. 2010.
- [17] E. Love, Y. Jin, and Y. Makris, "Proof-carrying hardware intellectual property: A pathway to trusted module acquisition," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 25–40, Mar. 2012.
- [18] A. Waksman and S. Sethumadhavan, "Silencing hardware backdoors," in *Proc. IEEE Symp. Security Privacy*, 2011, pp. 49–63.
- [19] M. Hicks, M. Finnicum, S. T. King, M. M. Martin, and J. M. Smith, "Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically," in *Proc. IEEE Symp. Security Privacy*, 2010, pp. 159–172.
- [20] C. Sturton, M. Hicks, D. Wagner, and S. T. King, "Defeating UCI: Building stealthy and malicious hardware," in *Proc. IEEE Symp. Security Privacy*, 2011, pp. 64–77.
- [21] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 296–310.
- [22] R. M. Rad, X. Wang, M. Tehranipoor, and J. Plusquellic, "Power supply signal calibration techniques for improving detection resolution to hardware trojans," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2008, pp. 632–639.
- [23] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *Proc. IEEE Int. Workshop Hardware-Oriented Security Trust*, 2008, pp. 51–57.
- [24] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, "Hardware Trojan horse detection using gate-level characterization," in *Proc. IEEE/ACM Design Autom. Conf.*, 2009, pp. 688–693.
- [25] Y. Alkabani and F. Koushanfar, "Consistency-based characterization for IC Trojan detection," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2009, pp. 123–127.
- [26] K. Hu, A. N. Nowroz, S. Reda, and F. Koushanfar, "High-sensitivity hardware trojan detection using multimodal characterization," in *Proc. IEEE Design Autom. Test Eur. Conf. Exhibit.*, 2013, pp. 1271–1276.
- [27] R. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1493–1502, Oct. 2009.
- [28] A. Kahng et al., "Watermarking techniques for intellectual property protection," in *Proc. IEEE/ACM Design Autom. Conf.*, 1998, pp. 776–781.
- [29] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *Proc. 16th USENIX Security Symp.*, 2007, pp. 291–306.
- [30] F. Koushanfar, I. Hong, and M. Potkonjak, "Behavioral synthesis techniques for intellectual property protection," *ACM Trans. Design Autom. Electron. Syst.*, vol. 10, no. 3, pp. 523–545, 2005.
- [31] A. Kahng et al., "Robust IP watermarking methodologies for physical design," in *Proc. IEEE/ACM Design Autom. Conf.*, 1998, pp. 782–787.
- [32] J. Lach, W. Mangione-Smith, and M. Potkonjak, "FPGA fingerprinting techniques for protecting intellectual property," in *Proc. IEEE Custom Integr. Circuits Conf.*, 1998, pp. 299–302.
- [33] G. Wolfe, J. L. Wong, and M. Potkonjak, "Watermarking graph partitioning solutions," in *Proc. IEEE/ACM Design Autom. Conf.*, 2001, pp. 486–489.
- [34] C. Alpert and A. Kahng, "Recent directions in netlist partitioning," *Integration, VLSI J.*, vol. 19, no. 1–2, pp. 1–81, 1995.
- [35] F. Koushanfar and Y. Alkabani, "Provably secure obfuscation of diverse watermarks for sequential circuits," in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2010, pp. 42–47.
- [36] A. Caldwell et al., "Effective iterative techniques for fingerprinting design IP," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 2, pp. 208–215, Feb. 2004.
- [37] J. B. Wendt, F. Koushanfar, and M. Potkonjak, "Techniques for foundry identification," in *Proc. Design Autom. Conf.*, 2014, DOI: 10.1145/2593069.2593228.
- [38] D. Holcomb, W. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.
- [39] Defense Advanced Research Projects Agency (DARPA), "Supply Chain Hardware Integrity for Electronics Defense (SHIELD)," Microsystems Technology Office/MTO Broad Agency Announcement, 2014.
- [40] F. Koushanfar and R. Karri, "Can the shield protect our integrated circuits?" in *Proc. Midwest Symp. Circuits Syst.*, 2014, pp. 51–57.
- [41] M. Rostami, M. Majzoobi, F. Koushanfar, D. Wallach, and S. Devadas, "Robust and reverse-engineering resilient puf authentication and key-exchange by substring matching," *IEEE Trans. Emerging Topics Comput.*, vol. 2, no. 1, pp. 37–49, Mar. 2014.
- [42] Y. Alkabani, F. Koushanfar, N. Kiyavash, and M. Potkonjak, "Trusted integrated circuits: A nondestructive hidden characteristics extraction approach," *Information Hiding*, ser. Lecture Notes in Computer Science, Berlin, Germany: Springer-Verlag, 2008, vol. 5284, pp. 102–117.
- [43] U. Ruhrmair, S. Devadas, and F. Koushanfar, "Security based on physical unclonability and disorder *Introduction to Hardware Security and Trust*," New York, NY, USA: Springer-Verlag, 2011.
- [44] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proc. IEEE/ACM Design Autom. Conf.*, 2012, pp. 83–89.
- [45] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 66–75, Jan./Feb. 2010.
- [46] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2007, pp. 674–677.
- [47] R. Chakraborty and S. Bhunia, "RTL hardware IP protection using key-based control and data flow obfuscation," in *Proc. IEEE Int. Conf. VLSI Design*, 2010, pp. 405–410.
- [48] R. Chakraborty and S. Bhunia, "Hardware protection and authentication through netlist level obfuscation," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2008, pp. 674–677.
- [49] R. Chakraborty and S. Bhunia, "Security against hardware trojan through a novel application of design obfuscation," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2009, pp. 113–116.
- [50] F. Koushanfar and G. Qu, "Hardware metering," in *Proc. IEEE/ACM Design Autom. Conf.*, 2001, pp. 490–493.
- [51] F. Koushanfar, "Provably secure active IC metering techniques for piracy avoidance and digital rights management," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 51–63, Feb. 2012.
- [52] F. Koushanfar, G. Qu, and M. Potkonjak, "Intellectual property metering," in *Proc. Inf. Hiding Workshop*, 2001, pp. 81–95.
- [53] Intelligence Advanced Research Projects Activity (IARPA), "Trusted integrated circuits program," 2011. [Online]. Available: <https://www.fbo.gov/utis/view?id=b8be3d2c5d5babddfc6975c370247a6>
- [54] R. Jarvis and M. G. McIntyre, "Split manufacturing method for advanced semiconductor circuits," U.S. Patent 7 195 931, 2004.
- [55] B. Hill, R. Karmazin, C. Otero, J. Tse, and R. Manohar, "A split-foundry asynchronous FPGA," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2013, DOI: 10.1109/CICC.2013.6658536.
- [56] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Proc. IEEE Design Autom. Test Eur. Conf. Exhibit.*, 2013, pp. 1259–1264.
- [57] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic encryption: A fault analysis perspective," in *Proc. IEEE Design Autom. Test Eur. Conf. Exhibit.*, 2012, pp. 953–958.
- [58] Chipworks, "Intel's 22-nm tri-gate transistors exposed," 2012. [Online]. Available: <http://www.chipworks.com/blog/technologyblog/2012/04/23/intels-22-nm-tri-gate-transistors-exposed/>
- [59] Defense Advanced Research Projects Agency (DARPA), "Integrity and reliability of integrated circuits (IRIS)," 2012. [Online]. Available: [http://www.darpa.mil/Our\\_Work/MTO/Programs/Integrity\\_and\\_ReliabilityofIntegratedCircuits](http://www.darpa.mil/Our_Work/MTO/Programs/Integrity_and_ReliabilityofIntegratedCircuits)
- [60] ExtremeTech, "iPhone 5 A6 SoC reverse engineered, reveals rare hand-made custom CPU, tri-core GPU." [Online]. Available: <http://tinyurl.com/9yn23he>
- [61] Chipworks, "Reverse engineering software." [Online]. Available: <http://www.chipworks.com/en/technical-competitive-analysis/resources/reverse-engineering-software>
- [62] Degate. [Online]. Available: <http://www.degate.org/documentation/>
- [63] W. M. V. Fleet and M. R. Dransfield, "Method of recovering a gate-level netlist from a transistor-level," U.S. Patent 6 190 433, 1998.
- [64] M. Hansen, H. Yalcin, and J. Hayes, "Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering," *IEEE Design Test Comput.*, vol. 16, no. 3, pp. 72–80, May/Jun. 1999.
- [65] W. Li, Z. Wasson, and S. Seshia, "Reverse engineering circuits using behavioral pattern mining," in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2012, pp. 83–88.

- [66] P. Subramanyan et al., "Reverse engineering digital circuits using functional analysis," in *Proc. IEEE Design Autom. Test Eur. Conf. Exhibit.*, 2013, pp. 1277–1280.
- [67] SypherMedia, "Syphermedia library circuit camouflage technology." [Online]. Available: <http://www.smi.tv/solutions.htm>
- [68] J. P. Baukus, L. W. Chow, R. P. Cocchi, and B. J. Wang, "Method and apparatus for camouflaging a standard cell based integrated circuit with micro circuits and post processing," U.S. Patent 2012 0 139 582, 2012.
- [69] J. P. Baukus, L. W. Chow, R. P. Cocchi, P. Ouyang, and B. J. Wang, "Building block for a secure CMOS logic cell library," U.S. Patent 8 111 089, 2012.
- [70] J. P. Baukus, L. W. Chow, and W. Clark, "Integrated circuits protected against reverse engineering and method for fabricating the same using an apparent metal contact line terminating on field oxide," U.S. Patent 2002 0 096 776, 2002.
- [71] J. P. Baukus, L. W. Chow, R. P. Cocchi, P. Ouyang, and B. J. Wang, "Camouflaging a standard cell based integrated circuit," U.S. Patent 8 151 235, 2012.
- [72] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proc. ACM Conf. Comput. Commun. Security*, 2013, pp. 709–720.
- [73] J. Rajendran, O. Sinanoglu, and R. Karri, "VLSI testing based security metric for IC camouflaging," in *Proc. IEEE Int. Test Conf.*, 2013, DOI: 10.1109/TEST.2013.6651879.
- [74] P. Rohatgi, "Improved techniques for side-channel analysis," *Cryptographic Engineering*. New York, NY, USA: Springer-Verlag, 2009, pp. 381–406.
- [75] C. Paar, J. Pelzl, and B. Preneel, *Understanding Cryptography: A Textbook for Students and Practitioners*. New York, NY, USA: Springer-Verlag, 2010.
- [76] F. Koeune and F.-X. Standaert, "A tutorial on physical security and side-channel attacks" *Foundations of Security Analysis and Design III*. Berlin, Germany: Springer-Verlag, 2005, pp. 78–108.
- [77] P. Rohatgi, "Electromagnetic attacks and countermeasures," *Cryptographic Engineering*. Berlin, Germany: Springer-Verlag, 2009, pp. 407–430.
- [78] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert, "Simple photonic emission analysis of AES," *J. Cryptogr. Eng.*, vol. 3, no. 1, pp. 3–15, 2013.
- [79] D. Genkin, A. Shamir, and E. Tromer, "RSA key extraction via low-bandwidth acoustic cryptanalysis," *Cryptology ePrint Archive*, Rep. 2013/857, 2013.
- [80] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerer's apprentice guide to fault attacks," *Proc. IEEE*, vol. 94, no. 2, pp. 370–382, Feb. 2006.
- [81] B. Yang, K. Wu, and R. Karri, "Scan based side channel attack on dedicated hardware implementations of data encryption standard," in *Proc. Int. Test Conf.*, 2004, pp. 339–344.
- [82] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic, "Securing designs against scan-based side-channel attacks," *IEEE Trans. Dependable Secure Comput.*, vol. 4, no. 4, pp. 325–336, Oct.-Dec. 2007.
- [83] M. Agrawal, S. Karmakar, D. Saha, and D. Mukhopadhyay, "Scan based side channel attacks on stream ciphers and their counter-measures," in *Proc. INDOCRYPT*, 2008, pp. 226–238.
- [84] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Side-channel analysis of PUFs and fuzzy extractors," *Trust and Trustworthy Computing*. New York, NY, USA: Springer-Verlag, 2011, pp. 33–47.
- [85] U. Rührmair et al., "Power and timing side channels for PUFs and their efficient exploitation," *Cryptology ePrint Archive*, Rep. 2013/851, 2013.
- [86] D. Karakoyunlu and B. Sunar, "Differential template attacks on PUF enabled cryptographic devices," in *Proc. Int. Workshop Inf. Forensics Security*, 2010, DOI: 10.1109/WIFS.2010.5711445.
- [87] A. Mahmoud, U. Rührmair, M. Majzoobi, and F. Koushanfar, "Combined modeling and side channel attacks on strong PUFs," *Cryptology ePrint Archive*, Rep. 2013/632, 2013. [Online]. Available: <http://eprint.iacr.org/>
- [88] B. Kopf and D. Basin, "An information-theoretic model for adaptive side-channel attacks," in *Proc. ACM Conf. Comput. Commun. Security*, 2007, pp. 286–296.
- [89] P. Rakers, L. Connell, T. Collins, and D. Russell, "Secure contactless smartcard ASIC with DPA protection," *J. Solid-State Circuits*, vol. 36, no. 3, pp. 559–565, Mar. 2001.
- [90] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," in *Proc. Eur. Solid-State Circuits Conf.*, 2002, pp. 403–406.
- [91] S. Moore, R. Anderson, R. Mullins, G. Taylor, and J. J. Fournier, "Balanced self-checking asynchronous logic for smart card applications," *Microprocess. Microsyst.*, vol. 27, no. 9, pp. 421–430, 2003.
- [92] F. Mace, F.-X. Standaert, I. Hassoune, J.-D. Legat, and J.-J. Quisquater, "A dynamic current mode logic to counteract power analysis attacks," in *Proc. Int. Conf. Design Circuits Integr. Syst.*, 2004, pp. 186–191.
- [93] M. Stanojlovic and P. Petkovic, "Strategies against side-channel-attack," in *Proc. Small Syst. Simul. Symp.*, 2010, pp. 86–89.
- [94] M. Joye, "Basics of side-channel analysis," *Cryptographic Engineering*. Berlin, Germany: Springer-Verlag, 2009, pp. 365–380.
- [95] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *J. Cryptogr. Eng.*, vol. 1, no. 1, pp. 5–27, 2011.
- [96] C. Clavier, J.-S. Coron, and N. Dabbous, "Differential power analysis in the presence of hardware countermeasures," *Cryptographic Hardware and Embedded Systems*, vol. 1965. Berlin, Germany: Springer-Verlag, 2000, pp. 252–263, ser. Lecture Notes in Computer Science.
- [97] P. C. Kocher, "Leak-resistant cryptographic indexed key update," U.S. Patent 6 539 092, 2003.
- [98] J. Demme, R. Martin, A. Waksman, and S. Sethumadhavan, "Side-channel vulnerability factor: A metric for measuring information leakage," in *Proc. IEEE Int. Symp. Comput. Architect.*, 2012, pp. 106–117.
- [99] J. Katz and V. Vaikuntanathan, "Signature schemes with bounded leakage resilience," *Advances in Cryptology*, vol. 5912. Berlin, Germany: Springer-Verlag, 2009, pp. 703–720, ser. Lecture Notes in Computer Science.
- [100] Y. Yu, F.-X. Standaert, O. Pereira, and M. Yung, "Practical leakage-resilient pseudorandom generators," in *Proc. ACM Conf. Comput. Commun. Security*, 2010, pp. 141–151.
- [101] F.-X. Standaert, O. Pereira, Y. Yu, J.-J. Quisquater, M. Yung, and E. Oswald, "Leakage resilient cryptography in practice," *Towards Hardware-Intrinsic Security*. Berlin, Germany: Springer-Verlag, 2010, pp. 99–134, ser. Information Security and Cryptography.
- [102] B. Yang, K. Wu, and R. Karri, "Secure scan: A design-for-test architecture for crypto chips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2287–2293, Oct. 2006.
- [103] D. Boneh, "Cryptography I," 2013. [Online]. Available: <https://class.coursera.org/crypto-007/class/index>.
- [104] L. Domnitser, N. Abu-Ghazaleh, and D. Ponomarev, "A predictive model for cache-based side channels in multicore and multithreaded microprocessors," *Computer Network Security*, vol. 6258. Berlin, Germany: Springer-Verlag, 2010, pp. 70–85, ser. Lecture Notes in Computer Science.
- [105] J.-S. Coron, P. Kocher, and D. Naccache, "Statistics and secret leakage," *Financial Cryptography*, vol. 1962. Berlin, Germany: Springer-Verlag, 2001, pp. 157–173, ser. Lecture Notes in Computer Science.
- [106] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *Proc. USENIX Security Symp.*, 2007, pp. 291–306.
- [107] V. Huard, M. Denais, and C. Parthasarathy, "NBTI degradation: From physical mechanisms to modelling," *Microelectron. Reliab.*, vol. 46, no. 1, pp. 1–23, 2006.
- [108] K. Chatterjee and D. Das, "Semiconductor manufacturers' efforts to improve trust in the electronic part supply chain," *IEEE Trans. Compon. Packag. Technol.*, vol. 30, no. 3, pp. 547–549, Sep. 2007.
- [109] S. Wei, A. Nahapetian, and M. Potkonjak, "Quantitative intellectual property protection using physical-level characterization," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1722–1730, Nov. 2013.
- [110] K. Huang, J. Carulli, and Y. Makris, "Parametric counterfeit IC detection via support vector machines," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, 2012, pp. 7–12.
- [111] X. Zhang, K. Xiao, and M. Tehranipoor, "Path-delay fingerprinting for identification of recovered ICs," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, 2012, pp. 13–18.

## ABOUT THE AUTHORS

**Masoud Rostami** received the M.S. degree in electrical engineering from Rice University, Houston, TX, USA, in 2010, where he is currently working toward the Ph.D. degree in computer engineering.

His research interests include double-gate devices, hardware security, and security of implanted medical devices.



**Farinaz Koushanfar** received the Ph.D. degree in electrical engineering and computer science and the M.A. degree in statistics, both from University of California Berkeley, Berkeley, CA, USA, in 2005.

She is currently an Associate Professor with the Department of Electrical and Computer Engineering, Rice University, Houston, TX, USA. Her research interests include adaptive and low-power embedded systems design, hardware security, and design intellectual property protection.



Prof. Koushanfar is a recipient of several awards and honors including the Presidential Early Career Award for Scientists and Engineers (PECASE), the ACM SIGDA Outstanding New Faculty Award, the NAS Kavli Foundation fellowship, and the young faculty (or CAREER) awards from the U.S. Army Research Office (ARO), the U.S. Office of Naval

research (ONR), the Defense Advanced Research Projects Agency (DARPA), and the National Science Foundation (NSF).

**Ramesh Karri** received the Ph.D. degree in computer science from the University of California at San Diego, La Jolla, CA, USA, in 1993.

He is a Professor of Electrical and Computer Engineering at the Polytechnic Institute of New York University, Brooklyn, NY, USA. He is the Area Director for cyber security of the NY State Center for Advanced Telecommunications Technologies at NYU-Poly; Hardware Security Lead of the Center for Research in Interdisciplinary Studies in Security and Privacy (CRISSP, <http://crissp.poly.edu/>), cofounder of Trust-Hub (<http://trust-hub.org/>), and organizer of the annual red team blue team event at NYU, the Embedded Systems Security Challenge (<http://esc.isis.poly.edu>). His research interests include trustworthy integrated circuits (ICs) and processors, high-assurance nanoscale IC architectures and systems, very large scale integration (VLSI) design and test, and interaction between security and reliability. He has over 150 journal and conference publications in these areas.



Prof. Karri was the recipient of the Humboldt Fellowship and the National Science Foundation (NSF) CAREER Award, and Best Student Paper Awards at the 2013 ACM Conference on Computer and Communications Security, the 2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, and the 2011 IEEE VLSI Design Conference.