Chip-Hong Chang · Miodrag Potkonjak
Editors

Secure System Design and Trustable Computing

Springer

Secure System Design and Trustable Computing

Chip-Hong Chang • Miodrag Potkonjak
Editors

Secure System Design and Trustable Computing

Springer

Editors
Chip-Hong Chang
School of EEE, Nanyang Technological University
Singapore

Miodrag Potkonjak
University of California Los Angeles
Los Angeles, CA, USA

ISBN 978-3-319-14970-7 ISBN 978-3-319-14971-4 (eBook)
DOI 10.1007/978-3-319-14971-4

Library of Congress Control Number: 2015945618

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Preface

Computer systems and internet-enabled smart electronic gadgets are increasingly integrated into our society. As a consequence of our heightened intimacy with this parallel cyberspace, security is becoming an integral part of any technology trend and affects stakeholders from all walks of life. Associated with the rise of social networking, consumerization of information technology, exponential growth of mobile devices and the many new services that leverage the omnipresence of Internet is a new battlefield between the attackers and defenders of information and privacy. The economic value and dependency on digital interconnectivity of people and "things" have come to the point that exploitation, even in a limited form, can be tremendously profitable. Malefaction can also be readily reinforced by technology to evade detection, bringing unique challenges to law enforcement of digital rights, protection of privacy, and combat against high-tech crimes. With attacks growing increasingly sophisticated, demand for better security will continue to drive technology innovation. As two transcendent topics, security and trust cut across many research areas and technological domains.

The aim of this book is to create an awareness of the ultimate challenges in addressing the security issues and vulnerabilities of embedded and integrated electronic systems, and to provide an insight into the emerging countermeasures and theoretical advancement in the design of secure and trustable computing circuits, systems, design platforms, and communication protocols. This book solicits and summarizes some key research topics and contributions in secure system design and trustable computing towards the goal of equipping the reader with sufficient background for further study of advanced topics. It consists of 17 chapters organized into three parts: Part I: Hardware Cryptographic Primitives, Part II: Hardware Counterfeiting and Integrity Protection, and Part III: Trust in Software, Networks and Services, with a deliberate emphasis on hardware-oriented security in view of the enormous attention and its rapid progress in recent years.

The rapid development of powerful embedded processors for mobile and embedded devices has endowed smartphones with the capability to serve as a unified platform for financial transactions, private information storage, and generation of authentication tokens for secure information exchange. Given the inherent mobility of such devices and their autonomy in handling the sensitive information, the threats of the device operating in an untrusted environment and the physical access to the system by the adversary are lifting. To replace the current best practice of using non-volatile and battery-backed random access memories for secret key storage, physical unclonable function (PUF) emerges as a lightweight and more vigilant entropy source and secret key generator in mobile system. The first three opening chapters of Part I are thus devoted to the overview, architectures, applications, design, and implementation of this promising cryptographic primitive. Specifically, the chapter "Disorder-Based Security Hardware: An Overview" introduces various examples of using disorder-based methods to avoid the long-term presence of keys in vulnerable hardware, and even completely evade the need for any security-critical information in hardware. This is followed by an extensive review of techniques proposed in the recent years for the design and implementation of high-quality PUFs for resource-constrained platforms in the chapter "Design and Implementation of High-Quality Physical Unclonable Functions for Hardware-Oriented Cryptography". The chapter "Digital Bimodal Functions and Digital Physical Unclonable Functions: Architecture and Applications" highlights the susceptibility of existing PUFs to environmental and operational variations and the difficulty of their integration into digital circuitry. A new notion of digital PUF is proposed to overcome these deficiencies. It exploits the time gap between the compact and expanded forms of digital bimodal function to realize public key communication and remote trust protocols.

For applications where the hardware authentication is performed by automatic identification technology such as magnetic stripes, barcode, smart cards, biometric identity cards, and RFID tags, there is a dire need for efficient on-chip implementation of public-key cryptosystem. It turns out that modular multiplication on data of very large word length is the most area-, time-, and power-consuming operation that determines to a large extent the overall cryptosystem performance. Due to the insurmountable carry-propagation of arithmetic in the accustomed weighted binary number system, the ancient Residue Number System (RNS) has resurged as a key player in boosting the cryptosystem performance. The chapter "Residue Number Systems in Cryptography: Design, Challenges, Robustness" provides a systematic and holistic treatment of the pivotal concepts of residue arithmetic and RNS applications in modern cryptography, from algorithm and complexity analysis to state-of-the-art hardware implementations. Efficient implementation of a cryptographic algorithm alone is inadequate as an attacker can perform a fault-based cryptanalysis to extract the embedded secrets if he has the physical access to a tamper-proof device to induce deliberate faults into it. The last chapter of Part I, "Fault Attacks on AES and Their Countermeasures" demonstrates an emerging field of powerful fault attacks known as Differential Fault Attacks (DFA) that can weaken modern ciphers like Advanced Encryption Standard (AES), and suggests an efficient Concurrent Error Detection (CED) scheme as an effective countermeasure.

Over the years, the geographical dispersion of chip design activities, coupled with the heavy reliance on third party hardware intellectual properties (IPs), has led to the infiltration of counterfeit and malicious chips into the integrated circuit (IC) design and fabrication flow due to the lack of effective mechanisms to detect and trace the use of semiconductor IPs. Reusable IPs sold in the form of high-level description language or Field Programmable Gate Array (FPGA) configuration bitstreams have been embedded into chips that control electronic systems and infrastructures of far greater value than their tangible worth. Unfortunately, these highly flexible expressions of hardware IPs are vulnerable to cloning, misappropriation and reverse engineering, and easily contaminated by hardware Trojans (HTs). Fabless semiconductor companies and field programmable device vendors that outsource their devices to merchant foundries for fabrication also find it difficult to prevent grey market sale of excess production, which can open out backdoors to catastrophic higher-level attacks on software, content, networks, and critical infrastructures. The consequence can be disastrous yet identifying compromised ICs is extremely difficult. New attack scenarios could put the integrated electronics ecosystem in dire peril if nothing is done to avert these hardware security treats. Part II addresses these problems. After the preamble of seven distinct categories of counterfeit electronics, the chapter "Circuit Timing Signature (CTS) for Detection of Counterfeit Integrated Circuits" discusses the use of Circuit Timing Signature (CTS) techniques to identify recycled, cloned, overproduced, and remarked ICs. The chapters "Hardware Trojan Detection in Analog/RF Integrated Circuits" and "Obfuscation-Based Secure SoC Design for Protection Against Piracy and Trojan Attacks" deal with HT detection in analog/RF and digital ICs, respectively. Using a wireless cryptographic IC as vehicle, the chapter "Hardware Trojan Detection in Analog/RF Integrated Circuits" demonstrates the effective field detection of HTs by combining side-channel fingerprinting with advanced statistical analysis and machine learning methods. In the chapter "Obfuscation-Based Secure SoC Design for Protection Against Piracy and Trojan Attacks", two different approaches based on the principle of design obfuscation are proposed to protect a System-on-Chip (SoC) design flow against HT attacks and IP piracy with minimal impact on the end-user experience. With the traditional finite state machine (FSM) synthesis procedure, there exist simple and effective ways to attack a sequential system and insert HT with negligible design overhead. The chapter "Towards Building Trusted Systems: Vulnerabilities, Threats, and Mitigation Techniques" identifies trust vulnerabilities in current industrial design tools and proposes a new practical approach to design trusted sequential system based on the FSM specification. The chapter "Hardware IP Watermarking and Fingerprinting" further elaborates the key ideas and dominating trends in hardware IP protection by watermarking and fingerprinting. It sheds light on the changing focus from watermark and fingerprint embedding and detection techniques to economically rewarding remote enforcement of semantic hardware and software IP rights. The chapters "IP Protection of FPGA Cores Through a Novel Public/Secret-Key Encryption Mechanism" and "Secure Licensing of IP Cores on SRAM-Based FPGAs" address the prospect of license violation and IP core over-deployment problems in FPGA IP market. Both chapters investigate mechanisms to assure the secure installation of FPGA IP cores onto contracted devices agreed upon by the IP provider and IP buyer through design encryption and public-key based authentication protocols in different perspectives of trustable design exchange scenarios.

As a consequence of the pervasiveness of computing devices, malicious software (Malware in short), such as worm, viruses, spyware, botnets, Trojan horses, rootkit, keyloggers, etc., have broadened their targets from desktop PCs and servers to smartphones and implantable medical devices. Malware penetration through cyberspace has threatened the economic growth and efficiency that a single, global interoperable network has brought us by bringing unique challenges to the security practitioners. Easy-to-use malware toolkits can automatically create hundreds of unique variants. The volume of new malware is growing at an exponential rate, amounted to an average of a million of new variants a day as reported by the Symantec Internet Security Threat Report (ISTR) in 2011. The security issues are further aggravated by the cloud computing paradigm, Online Social Network (OSN) and Mobile Adhoc Network (MANET) that make offloading of computing tasks, locating and collection of unprecedented amount of private information easier than ever. Part III is devoted to the problems and solutions pertaining to the safe deployment and trustworthiness of software, distributed networks, services, and infrastructures. The chapter "Heterogeneous Architectures: Malware and Countermeasures" discusses the threat of malicious software on heterogeneous platforms and explains how advanced virtualization technology can monitor, analyze, and protect heterogeneous software and hardware architectures from malware attacks. The chapter "Trusted, Heterogeneous, and Autonomic Mobile Cloud" surveys state-of-the-art research on theoretical advancement and practical implementations of trusted computing on mobile cloud, featuring different levels of resiliency against malicious and misbehaved nodes. The chapter "Infiltrating Social Network Accounts: Attacks and Defenses" looks into the infiltration attacks on social network accounts. Venue centric location verification solutions that use certified location information are proposed for privacy protection in cyberspace social networking. The chapter "An Economical, Deployable and Secure Architecture for the Initial Deployment Stage of Vehicular Ad-Hoc Network" explores economical, deployable, and secure vehicular ad-hoc network (VANET) system design to facilitate the gradual deployment of wireless communication among vehicles. Workable models for multi-confidence level data verification and time-location based secure positioning systems are established to achieve connectivity with a high degree of confidence using only a small number of smart vehicles and economical roadside units that do not need expensive internet access. Critical systems like VANET demand more than just fault tolerance and security. They must be able to continue delivering essential services during attacks, faults, or accidents. The last chapter "Deception-based Survivability" of this book examines a new generation of defense systems that are smart, adaptive, and possess deception-based survivability to make them stay ahead of the malicious actors. It presents a high-level deployment architecture that uses deception to trace attacker intent, objective and strategies (AIOS) for the development of targeted recovery and adaptation procedures.

The threat landscape has become increasingly complex with the pervasiveness of smart devices, immense data, extensive virtualization, and dense connectivity. It is impossible to cover all aspects of attack scenarios and defence methodologies in one edited book volume. We would like to thank the expert researchers who have contributed to the broad spectrum of insightful techniques and solutions presented in this book. With supporting motivation and background material, it is our hope, that the historical notes, rigorous treatment, breadth and extensive bibliography will make this book a comprehensive source and an important reference for the design of security and trustable computing systems, that serves students, instructors, and research professionals in the community. It is our hope that this edited book volume will facilitate further advancement of the field and that we have helped play a small part in it.

Singapore — Chip-Hong Chang
Los Angeles, CA, USA — Miodrag Potkonjak

Contents

Part I
Hardware Security Primitives

Disorder-Based Security Hardware: An Overview

Ulrich Rührmair

Abstract The explicit utilization of physical disorder and of random, micro- and nanoscale phenomena is a recently emerging trend in hardware security. The associated fields of research could be termed *disorder-based security* or also *nano-security*. In this chapter, we give an overview of this arising area. We start by a motivation of alternative approaches in hardware security. This is followed by a brief description of physical disorder and its useful features.

Subsequently, readers are introduced to the main concepts of the area via a number of concrete examples. We show how disorder-based methods can avoid the long-term presence of keys in vulnerable hardware, allow the derivation of keys in systems without non-volatile memory, and sometimes even evade the need for any security-critical information in hardware at all. Our examples include optical as well as electrical implementations.

Towards the end, we take a broader perspective of the field: We illustrate its history from its first presence in patent writings in the 1960s over the pivotal role of physical unclonable functions (PUFs) to its current state and future challenges.

1 General Context and Chapter Overview

Several of the earliest documented examples of cryptographic techniques are owed to the Greek historian Herodotus. Among others, he describes an ancient case of a steganographic technique in the conflict between Persia and Greece around 500 BC: In order to communicate sensitive information, the Greek tyrant Histiaeus shaved the head of a slave and tattoed a confidential message onto the scalp. Once the hair had re-grown, the slave could serve as a secret message carrier, passing adversarial territory unrecognizedly [24]. Herodotus also reports that around the same time, the Spartans encoded their military messages by use of a wooden stick of a well-defined diameter. A leather belt was wrapped around the stick, and the message was written

U. Rührmair (✉)
Horst Görtz Institute for IT-Security, Ruhr University Bochum, Germany
e-mail: ruehrmair@ilo.de

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_1

across the bends. Without the stick, the symbols appeared randomly distributed over the belt; but by winding it around a stick with the same diameter, the message could be recovered [160].

Said two techniques are, to our knowledge, the first documented methods that explicitly use physical and even biological phenomena for information protection. Nothing else holds for the newly emerging fields of "*disorder-based security*" or "*nano-security*" that we discuss in this chapter—with the difference, however, that the focus is now on phenomena at the micro- and nanoscales, and on the explicit utilization of physical disorder at these minuscule scales.

This chapter provides a first introduction and overview of these recent and fast-moving areas: Sect. 2 motivates why alternative approaches in security are useful and sometimes necessary. Section 3 discusses the general phenomenon of physical disorder and its usefulness. Section 4 provides readers with three central application examples. We thereby didactically take a inductive approach, introducing readers to the central concepts in the area by concrete examples. Section 5 summarizes concrete advantages of disorder-based methods. Section 6 then takes a broader, partly historic perspective of the field. We conclude by a summary in Sect. 7.

2 Why Investigate Alternative Approaches?

Why should one investigate alternative approaches to hardware security at all? Why should one be bothered to investigate the complex phenomena of physical disorder and randomness? We give some of the main reasons below.

2.1 Vulnerabilities of Classical Secret Keys

In agreement with Kerckhoffs' principle [56], most current cryptographic and security methods rest on the concept of a secret key. This forces security hardware to permanently store a bit string that is unknown to the adversary. This requirement can be unexpectedly difficult to realize in practice: On the physical level, invasive techniques, semi-invasive methods and side channel attacks can be used to extract valuable key information [2]. On the software side, malware like Trojan horses or viruses may read out and transfer keys, even without the notice of users [2].

Three aspects play into the hands of attackers in this context. Firstly, keys stored in non-volatile memory (NVM) are permanently present in the hardware system in a relatively easily accessible digital form [2]. What makes things worse, the permanent storage can even leave traces in the memory that allow recoverage of the key *after* it has been erased [42, 140, 161]. Secondly, keys are mostly strings with high entropy, allowing their identification within other, less entropic data in computer memory relatively easily [135]. Finally, the requirement that modern hardware should be lightweight, mobile, and inexpensive often leaves little room for sophisticated key protection. Functionality and cost aspects dominate security requirements in many commercial scenarios [2].

Ron Rivest accurately subsumed the situation in a keynote talk at Crypto 2011 by commenting that "*calling a bit string a secret key does not make it secret, but rather identifies it as an interesting target for the adversary*" [92]. This makes effective key protecting mechanisms—or better still: methods to entirely avoid classical secret keys in vulnerable hardware—an important research topic.

2.2 Practicality and Cost Aspects

There is one second issue of classical methods. As R. Pappu et al. put it in their seminal article on Physical One-Way Functions in SCIENCE MAGAZINE [89]: "*Cryptosystems don't protect information if they're not used.*" Indeed, the classic implementation of secret key schemes in hardware makes two implicit assumptions: Firstly, that the security hardware contains non-volatile memory (NVM) cells, in which the key can be stored. Secondly, that it has sufficient computational capacities to implement the cryptographic schemes which process the key.

Both assumptions are not met in certain situations. To start with, not all security-relevant hardware contains NVM cells. This includes central processing units (CPUs), several types of field programmable gate arrays (FPGAs), certain lightweight security systems, etc. If the keys are stored in a second, accompanying piece of hardware (for example the computer's hard disk), the transfer of the key to units where the key is needed (CPU, FPGA, etc.) creates an explicit attack point. This is obviously disadvantageous—a self-contained security solution would be preferable.

Secondly, in several very low-cost scenarios, the security hardware does not possess extensive computational capacities. As an example, consider the forgery-proof tagging or "labeling" of valuable objects, such as branded products, electronic components, valuable documents, and the like. There is no computational capacity in a Rolex watch, a Nike shirt, or a paper document. Adding such capacity by RFID tags may be too expensive, apart from the obvious privacy problems it creates. Still, as pointed out by Kirovski, 7–10 % of the world trade consists of forged products, causing an overall economic loss of the order of hundreds of billions of dollars [59].

The example illustrates the existence of highly relevant security problems that are difficult to address by standard techniques.

3 Physical Disorder and Its Useful Features

The discussion of the last section motivates the search for other methods. This chapter indeed provides an overview of an alternative approach that has emerged over the last decade(s), so-called "*disorder-based security*" or "*nano-security*". This area

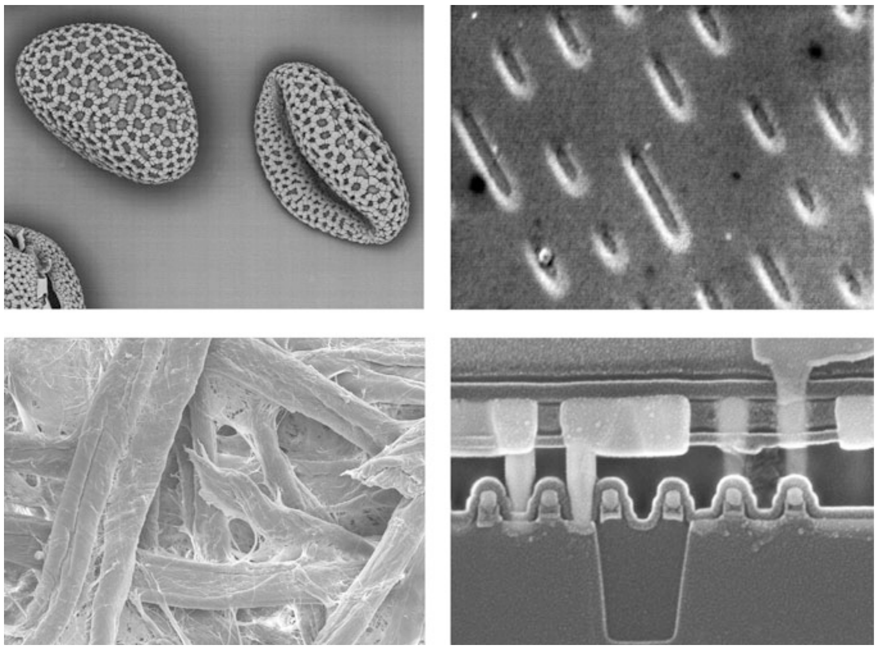


Fig. 1 Microscopic images of several everyday objects: Pollen of Lilium auratum (*upper left*) [132], ordinary (filter) paper (*bottom left*) [133], a customary CD (*upper right*) [134], and a cross section through Apple's A5 chip (*bottom right*) [51]

explores how small-scale, random physical disorder and hardware imperfections can be exploited advantageously in security and cryptography.[1] Said physical disorder is actually omnipresent in our everyday world: Essentially all physical systems exhibit it intrinsically and "for free" on small enough length scales.

Four illustrating examples are given in Fig. 1: Firstly, many biological structures show fascinating small-scale irregularities, in our example pollen of the Lilium auratum (top left). Also customary paper exhibits notable three-dimensional randomness, for example in its interwoven "*paper fibers*"; our image shows a close-up of customary filter paper (bottom left). Thirdly, modern integrated circuits exhibit complex manufacturing variations as well. These do not affect their digital functionality, but still notably influence their exact analog properties, for example the runtime delays in their individual components (bottom right). Finally, also storage media such as compact discs are subject to imperfections, for example in the exact shape and length of their information-carrying indentations. The deviations

[1]This focus distinguishes the area from other, well-known and non-standard approaches in crypto and security, such as quantum cryptography [8], noise-based crypto [25], or the bounded storage model [5, 79].

are too small to affect the stored content, but still constitute a unique sub-structure of each disc (top right).

What are the features that makes physical disorder useful in cryptographic and security applications? At least four of them are discussed below.

Omnipresence

As already emphasized, essentially all physical objects exhibit a certain amount of random disorder at sufficiently small length scales.[2] This phenomenon is not limited to the examples of Fig. 1. Readers may take a short virtual tour through their offices: Any chairs, tables, walls, windows, pens, etc. exhibit small-scale disorder and imperfections, be it due to their production process, wear and tear, or both.

In fact, it is very difficult to imagine a macroscopic system or production process that is disorder-free. Usually, this is regarded as disadvantageous, for example in the context of semiconductor fabrication or nanofabrication. The approaches presented in this thesis turn the inevitability of disorder into an advantage, though, exploiting it for hardware identification, secret key derivation, and other security applications.

Hard to Clone

A second central feature of physical disorder is that it is infeasible to perfectly clone it with current fabrication technology.[3] This is often referred to as (physical) "*unclonability*" in the literature [36, 89] and in this thesis. Interestingly, the unclonability of a system may still hold even if its entire structure is known down to every single atom to an attacker. Note that in the physical world, *knowing* the structure of a system and *rebuilding* it accurately are two different things. Consider the paper surface as of Fig. 1 as an example: Even if the exact position of all paper fibers would be known, it would still remain prohibitively difficult to refabricate it perfectly. This physical feature stands in sharp contrast to the conditions in a mathematical or Turing machine world: If you know a bitstring exactly, it is trivial to

[2]The only macroscopic or mesoscopic counterexamples known to the author are highly regular crystal structures, but even they can exhibit defects or surface roughness. In addition, there are certain *microscopic* objects like photons or electrons which appear to be the same for every specimen (compare [162] for an amusing assessment of the similarities of all electrons by two great physicists). But such *microscopic* objects or even elementary particles are not our topic in this thesis.

[3]Please note that this type of unclonability differs from another well-known type of unclonability, namely quantum unclonability. The latter is based on inherent features of quantum mechanics, the former on the technological limitations of available two- and three-dimensional fabrication techniques.

copy it with perfect accuracy. The underlying physical phenomenon could be termed *(re-) fabrication complexity*, in analogy to the well-known term computational complexity that underlies mathematical cryptography.

Hard to Fully Characterize

Disordered systems can possess a very large entropy or random information content. As an example, consider the random information contained in the random microscopic structure of a A4-sized sheet of paper (see again Fig. 1). It is infeasible to completely measure (i.e., to "characterize" in physical parlance) this information with current technology in short time. At the same time, the generation of this disorder is very inexpensive, occurring as a natural byproduct in the fabrication process. This points to a certain *asymmetry* in the physical world between *generating* randomness and *measuring* it. Again, this asymmetry has no direct analog in the Turing world: On a Turing machine, reading a bit from the tape and generating a random bit on the tape take essentially the same effort. The associated physical phenomenon could be termed *measurement complexity* or *characterization complexity*, again analog to the well-known computational complexity.

Hard to Simulate on a Turing Machine

Simulating the input-output behavior of complex, disordered physical structures on a Turing machine can be laborious. A straightforward example are the interference patterns created by disordered optical systems upon laser illumination (see [88, 89] and Sect. 4.3), but also electrical and quantum systems with similar properties exist [7, 26, 31, 33, 94, 96]. One reason for the observed simulation overheads are the inherently parallel and analog interactions in solid state systems. They are usually expensive to emulate on digital, sequential computers. This usually makes the digital simulation of a given physical system notably slower than the systems' real-time behavior, and can even render such simulation *practically infeasible* at all (compare [33]). The associated phenomenon could be called the *simulation complexity* of physical structures. Interestingly, the presence of disorder is no necessary prerequisite of simulation complexity, since also quantum systems may be hard to emulate. But in the classical physical systems that are considered in this chapter, the occurrence of complex disorder usually increases the simulation overhead.

Similar to our above discussion, the phenomenon of simulation complexity has no direct counterpart in mathematical cryptography. In our sense, it can only emerge when two different worlds, like the physical and the Turing world, and their "computational speeds" are compared to each other.[4]

Simulation complexity can be utilized in different ways: Firstly, it may render the simulation of certain disordered structures too complex to be practically feasible at all (compare Pappu et al. [89] and Sect. 4.3). Secondly, simulation may be possible in practice, but notably more time consuming than the real-world behavior of the disordered structure. The latter is explicitly exploited by the recently emerging primitives of "*SIMPL systems*" [94] and "*Public PUFs*" [7].

How to Utilize Physical Disorder?

Given the above discussion, it seems almost straightforward to exploit physical disorder in a security context. Just to name two examples: Why not derive unforgeable "*fingerprints*" for all everyday objects, valuable products, and security items from their individual surfaces? Why not straightforwardly derive internal secret keys from the individual disorder that is present in every piece of silicon hardware, and identify this hardware via this key? However, as common in scientific research, the problems and scientific challenges lie in the details. Certainly all everyday objects exhibit disorder on small length scales, ultimately when being scanned with an (expensive) atomic force microscope. But which features can be measured particularly inexpensively, are stable over time, and are still most difficult to forge or imitate? Which nanostructures and materials lead to particularly secure and practical fingerprints? How can honest users know the "correct" fingerprints of authentic objects, as opposed to the fingerprints of unauthentic objects? Etc.

Around these questions, a rich research landscape has emerged within the last years [71, 109]. It spans from nanophysics and electrical engineering to theoretical computer science and mathematics, and is concerned with implementational questions as well as with the theory behind disorder-based security. Some main examples are discussed throughout the rest of this book chapter.

[4]It is interesting to comment that any physical action can in principle be interpreted as a computation and, vice versa, that computation can be understood as an inherently physical process. This view has been expressed by Deutsch and others [31, 32, 143], and, in a non-scientific context, even a few years before Deutsch by novelist Douglas Adams [1]. In this sense, it appears legitimate to talk about "computational speed" also when one is actually referring to physical interactions, as we do above.

4 Examples of Disorder-Based Security Methods and Hardware

We will now illustrate the practical usability of physical disorder by three concrete examples. Among other things, our discussion details the concrete security advantages of the examples over classical techniques.

4.1 Certificates of Authenticity from Paper Irregularities

According to Kirovski [59], it is estimated that 7–8 % of world trade, 10 % of the pharmaceutical market, and 36 % of the software market consist of counterfeit products, causing a loss of hundreds of billions of US-Dollars every year [59]. This calls for inexpensive and effective methods that verify the authenticity of products and other objects of value. Ideally, one would like to set up a system where certain "*certification authorities*", for example product manufacturers or state authorities, can create unforgeable "*certificates of authenticity (COAs)*" for valuable objects [59]. The COAs should be machine readable, and should be verifiable by a large number of widespread "*testing devices*" [59]. Ideally, but not necessarily, the latter might be handheld and owned by security-aware consumers themselves.

Since paper is a very widespread material, it seems suggestive to utilize the random and unclonable structure of paper in this context (compare Fig. 1). Recall that the latter induces an individual fingerprint of any paper medium, including paper documents, paper packages, and paper banknotes. Approaches in this direction have indeed been suggested by a number of researchers in the past [14, 40, 43, 59, 141]. We describe their technique by the example of paper banknotes below.

Protocol 1: CERTIFICATES OF AUTHENTICITY (COAS) FOR PAPER BANKNOTES

Set-Up Assumptions:

1. The banknote manufacturer (BM) holds a secret signing key SK from some cryptographic digital signature scheme.
2. All testing devices (TDs) hold the public verification key VK that corresponds to SK.
3. The BM has implemented a physical method to measure the random structure of a given paper surface. The method produces a compact digital strong $\mathsf{UF(S)}$ describing the structure.[5]
4. All TDs have implemented a similar method and can reproduce the measurement results of the BM in a reliable fashion. That is, given the same piece of paper S as the BM, each TD will derive the same description $\mathsf{UF(S)}$, within some error thresholds.

[5]One advantageous approach is shining a laser beam at the structure and measuring the resulting reflective interference pattern [14], but there are also other suitable techniques [43, 155, 156].

This presumes that the measured paper features are sufficiently stable against wear-and-tear and aging.

COA Generation:

1. The BM fabricates a paper banknote. It measures the random paper structure in a selected, marked subregion S of the banknote, producing a digital string $\mathsf{UF(S)}$ that describes the structure.
2. The BM creates a digital signature $\mathsf{DigSig_{SK}(UF(S), I)}$, and prints the information

$$\mathsf{UF(S),\ I,\ DigSig_{SK}(UF(S), I)}$$

onto the bank note, for example via a two-dimensional barcode.

Thereby I can be an arbitrary accompanying information, for example the banknote's value, its printing date and place, etc.

The unit consisting of an unclonable physical structure S, a digital string $\mathsf{UF(S)}$ that describes the unclonable features of S, and a digital signature $\mathsf{DigSig_{SK}(UF(S), I)}$, is then termed a "COA" in our sense [59].
COA Verification:

1. The TD reads the information $\mathsf{UF(S)}$, I, $\mathsf{DigSig_{SK}(UF(S), I)}$ from the banknote.
2. The TD verifies the validity of the digital signature $\mathsf{DigSig_{SK}(UF(S), I)}$ by use of its verification key VK.
3. The TD measures the random paper structure of the banknote, and checks if the results match the information $\mathsf{UF(S)}$ printed on the banknote, again within some error thresholds.
4. If the tests in step 2 and 3 are passed, the TD regards the banknote as genuine.

4.1.1 Security Discussion

The above scheme is secure under the following assumptions:

- The adversary cannot gain access to the secret signing key stored at the manufacturer.
- The digital signature scheme is secure.
- The adversary cannot clone the paper structure, i.e., he cannot fabricate any physical system that "looks" like the original paper within the accuracy limits of the applied measurement method.

It is not too difficult to see that all of these assumptions are also necessary: If the adversary has access to the signing key, he can create COAs by himself. The same holds if the digital signature scheme is insecure, and if the adversary can forge signatures for any given plaintext. Thirdly, if the adversary can clone the paper in the above sense, he can fake notes by (1) copying the paper structure of a given banknote, and by (2) using the very same digital signature from this note on the new, forged note.

4.1.2 Potential Advantages and Drawbacks

What are the advantages and disadvantages of the above approach? One notable upside lies in the way it treats secret keys. Astonishingly, there is no secret key or other security-critical secret information on the banknotes/COAs. One could allow an adversary to inspect every atom of the banknote, and still the COAs could not be forged: Knowing the paper structure and physically reproducing it are two different things. Even the testing devices do not need to contain any security-critical information, since the adversary does not benefit from learning the public key that is stored in them! Both features particularly shine in applications where adversaries can easily gain long-term access the COAs (including banknotes), or whenever there are many, widespread testing devices that can potentially be accessed by adversaries (for example widespread devices in retail stores, supermarkets, pharmacies, etc.).

The only secret key of the entire scheme rests in the hands of the manufacturer, where it can usually be very well protected. There is one further noteworthy aspect in this context: Think about a scenario with many decentral fabrication sites, all of which manufacture products that need to be equipped with COAs. It seems that all of these sites would then need to be equipped with a secret signing key, potentially creating security gaps. However, some thinking shows that indeed all necessary digital signatures could even be created centrally by one authority, and later be distributed to the sites. The sites merely send the strings UF and perhaps I to the central authority, which returns the signature $\mathsf{DigSig}_{\mathsf{SK}}(\mathsf{UF}(\mathsf{S}), \mathsf{I})$. Without going into the details, we remark that such an approach could be well applied against of gray-market IC overproduction, in particular whenever IC fabrication is outsourced. While the design is given to external manufacturers abroad, the certification process and signing key remains under the full control of the IP owner.

There is a third security upside of the above COAs. Standard security features of banknotes can be mass produced by the right printing equipment. In other words, the technology to produce many identical specimen exists already; if the adversary gains access to it, he will succeed. The security of current banknotes hence rests on the assumption that fraudsters will not gain access to a certain, existing technology. To the contrary, currently no technology exists that could exactly clone the complex, three-dimensional structure of paper. Even if developed some day, it would likely not immediately lend itself to cost-efficient mass fabrication. This creates an extra security margin against fraudsters, some sort of "*technological security*", as opposed to the access security assumption of traditional money printing.

Readers well-versed in mathematical cryptography may object at this point: Can digital signatures provide the long-term security required in banknotes? Recall that schemes with fixed key length may become insecure after a few decades [15].

However, there are a few counterarguments to this objection. Firstly, banknotes are steadily exchanged in relatively short intervals. According to information provided by the Deutsche Bundesbank and Giesecke and Devrient [117], for example, all German banknotes are exchanged every one to five years. The newly printed notes could use longer signature keylengths, steadily adjusting security. Similar considerations hold for archival uses of COAs where long-term security is a necessity, such as birth certificates: The digital signatures could be "refreshed" by techniques well-known in the community [147]. Finally, careful choice of keylengths and elliptic curve schemes may already in itself provide strong long-term guarantees, as detailed in [65].

Let us conclude this discussion by looking at some practical aspects. While our approach offers strong security advantages, the cost and practicality aspects are rather mixed. On the upside, it becomes unnecessary to attach dedicated labels to the banknotes, creating some cost advantages. On the other hand, extra costs in the production process are generated: The random paper structure needs to be measured, the signature must be generated, and individualized information has to be printed on each note. Furthermore, verification potentially requires a costly measurement device, for example in order to position the banknote very accurately and re-generate the original measurement value. This can partially eat up the cost savings gained by avoiding dedicated labels.

4.1.3 Variants

The above COA-technique based on digital signatures and unclonable structures has manifold variations. Firstly, one can attach dedicated, tailor-made unclonable structures ("labels") to the valuable objects, instead of exploiting intrinsic features of the objects themselves. This partly creates extra cost. But at the same time, it can increase unforgeability, and may make the measurement process more efficient and inexpensive. The reason is that dedicated, tailormade labels can have extra secure and more easily measurable unique features. Various dedicated unclonable structures have been suggested to this end; see, e.g., [14, 17, 21, 22, 29, 44, 57, 58, 136, 158], and references therein.

Secondly, COAs can be used for content protection [40, 44, 53, 158]. The key observation here is that storage media may have random, unclonable features, too. For example, the small-scale structure S of the lands and pits of a CD is subject to manufacturing variations, and thus exhibits unique features $\mathsf{UF}(\mathsf{S})$ [44, 158]. Creating a digital signature $\mathsf{DigSig}_{\mathsf{SK}}(\mathsf{UF}(\mathsf{S}), \mathsf{I})$, where I contains a hash value of the digital content stored on the CD, links this very content to its unique storage medium, thus certifying it. Copying the content onto another data carrier invalidates this certificate. Similar considerations hold for content printed on paper, such as business contracts, as discussed in [21, 40, 136].

4.2 Secret Cryptographic Keys from SRAM Power-Up States

Secret keys are at the heart of most modern cryptographic and security schemes. But as mentioned earlier, storing them in hardware can be non-trivial: On the security side, powerful attacks have been developed, ranging from invasive to side channel techniques [2]. On the cost/practicality side, not every hardware system contains NVM cells for storing secret keys.

An alternative approach, which can potentially improve both on security and practicality, is to exploit physical disorder, more precisely the random and individual manufacturing variations in each hardware system, as a secret key source. Perhaps the most prominent example for this technique are SRAM cells: Upon power-up, each cell contains either a zero or one, depending on the random manufacturing variations present in the cells [41, 49, 50]. The power-up states are relatively well repeatable upon multiple power-ups for each single cell, but they statistically vary almost uniformly from cell to cell in an SRAM array. The k cells in an SRAM array thus together create an individual power-up state "fingerprint", allowing derivation of an individual key. In the parlance of the field, an SRAM cell can act as a so-called "*physical unclonable function*" or "*PUF*", leading to the widespread terminology "SRAM PUF" for the above phenomenon [41].

If this approach is used in practice, there must not be a single bit flip in the derived secret key. Error correction (EC) therefore is vital, since not all states are perfectly stable upon multiple power ups. Most EC techniques thereby have in common that some public, non-secret "helper data" or "error-correcting data" is provided to the hardware system, allowing derivation of a stable key from the noisy power-up states [41]. it is well-known that this error-correcting helper data can be constructed in such a way that the helper data alone—i.e., without knowledge of the power-up values of the SRAM cells—does not leak any knowledge about the derived key. It is interesting to observe that the use of helper data shifts the problem of storing data permanently: Instead of a binary key, now the helper data needs to be stored in NVM. One difference is, though, that the helper data can be stored publicly, since it does not leak information about the key. It needs to be provided to the secure hardware only whenever key derivation is necessary.

An illustrative example application of the above phenomenon and of "*SRAM PUFs*" is the protection of intellectual property (IP) of FPGA designs [41]. Many FPGA types do not contain non-volatile memory cells and hence cannot store application designs permanently [41]. The designs are thus put in external memory and uploaded onto the FPGA when needed. This has the disadvantage that fraudsters can intercept the uploaded bitstream and learn the designs, which often represent a very substantial IP value. In principle, the bitstream could be encrypted, but the lack of NVM on the FPGA prevents the storing of classical secret keys on the FPGA. At the same time, however, SRAM cells are present on many FPGAs, and their power-up states can be used to derive a key. This enables IP protection schemes between the manufacturer, the FPGA, and an external memory device storing the design [41]. We give one basic example of such a scheme below [41]; other, more involved techniques are described in the same reference [41].

Protocol 2: IP PROTECTION OF FPGA DESIGNS VIA SRAM PUFS [41]

Set-Up Assumptions:

1. The scheme involves four parties: The IP provider (IPP); a system integrator or designer (SYS); the FPGA-manufacturer (HWM); and a trusted third party (TPP).
2. The communication channels between HWM and TTP, and between TTP and IPP, are authenticated and confidential.
3. The TTP and the HWM are fully trusted.
4. The HWM can disable access to the SRAM cells after reading them out in the enrollment phase (for example by blowing some fuses). No one can access the cells anymore after this operation, including adversaries.
5. For simplicity of exposition, we do not explicitly deal with error correction in this protocol. In practice, error correcting helper data does need to be used to obtain stable responses.[6]

Initialization Phase (aka "Enrollment Protocol" [41]):

1. The HWM associates an $\mathsf{ID_{HW}}$ to a given FPGA. It reads out different sets of SRAM-power up states $R_1, \ldots, R_n$ of this FPGA.
2. The HWM disables external access to those SRAM-cells that have provided the above response $R_1, \ldots, R_k$. Internal access for the FPGA itself to these cells must remain intact.
3. The HWM sends

$$\mathsf{ID_{HW}}, R_1, \ldots, R_n$$

   to the TTP.

IP Authentication Protocol:

1. SYS sends

$$\mathsf{ID_{SW}}, \mathsf{ID_{HW}}$$

   to TTP, indicating which software $\mathsf{ID_{SW}}$ shall be utilized on which FPGA hardware $\mathsf{ID_{HW}}$.
2. The TTP sends $\mathsf{ID_{SW}}$ to the IPP, and the IPP returns the software SW to the TTP.
3. The TTP encrypts the software with the key R_i, creating a value

[6]Following a convention stipulated in [41], readers may interpret the protocol in such a way that C_i denotes the PUF challenge *and* the corresponding helper data required to reconstruct the PUF response R_i from a noisy version R_i'.

$$D = \mathsf{Enc}_{R_i}(\mathsf{SW}, \mathsf{ID}_{\mathsf{SW}}).$$

4. The TTP sends the message

$$C_i, C_j, D, \mathsf{MAC}_{R_j}(C_i, C_j, D)$$

   to SYS.

Design Upload and Decryption on the FPGA:

1. The FPGA uploads the encrypted bitstream created by SYS, which is stored in a (non-confidential) storage medium accompanying the FPGA.

   The bistream potentially contains k encrypted and authenticated software blocks of the above form

$$C_i^k, C_j^k, D^k, \mathsf{MAC}_{R_j^k}(C_i^k, C_j^k, D^k).$$

2. For each k, the FPGA internally reproduces the responses R_i^k and R_j^k by accessing and measuring the respective SRAM cells.

   (Please note again that in practice, error correcting helper data must be used to this end, which must be provided from and external non-volatile, but not confidential storage medium. In the case of FPGAs, the same medium can be used that stores the encrypted upload bitstream.)

   The FPGA decrypts the bitstream and verifies the authentication.
3. The FPGA is configured by the decrypted bitstream.

4.2.1 Security Discussion

Our security discussion below follows [41]. The protocol's aim is to achieve confidentiality and integrity of the software blocks and thus of the related IP. It achieves this aim under the following assumptions:

- The TTP and the HWM are trusted.
- The mutual communication channels TTP-HWM and TTP-IPP are confidential and authenticated.
- No adversaries can externally read-out the responses R_i and R_j after access to them has been disabled by the HWM, while the FPGA itself can still access the responses internally.

We comment that these are relatively strongly assumptions. In particular, the third hypothesis is at the least in part comparable to the standard assumption that a classical key cannot be read-out by the adversary.

4.2.2 Potential Advantages and Drawbacks

Perhaps the main advantage of the above scheme is that it enables security (and encryption) in an environment without NVM. Without using SRAM cells as key source, no encryption would be possible at all. This could be regarded as a practicality advantage or as a security advantage, depending on personal taste.

It has also been argued that the use of SRAM PUFs brings about general security advantages in comparison with NVM cells, i.e., even in comparison with systems that do possess NVM. Such claims require further analysis, we believe. It is true that SRAM cells allow to derive a key only whenever it is needed within the hardware. This means that the key is not present permanently in the system in more or less digital form, as in the case of NVMs. On the other hand, invasive or other access to the SRAM cells [84] does allow derivation of the key, just as in the case of NVMs. Furthermore, also cloning of SRAM PUFs has been reported recently [45]. Overall, we recommend that the exact security gains of using SRAM PUFs over NVMs should be analyzed separately and carefully for any given system and application by its users.

One potential drawback of the approach is that SRAM PUFs require error-correcting helper data for deriving a stable key. This needs to be provided either from external, non-volatile memory, or from a trusted third party during a certain protocol. On the other hand, as mentioned earlier, the helper data can be constructed in such a way that it does not leak any information (in an information-theoretic sense) about the key, as long as the power-up states of the SRAM cells are unknown to an attacker. This means that in opposition to a classical key, the helper data at least does not need to be kept secret.

4.2.3 Variants

Every communication of the above scheme runs over the TTP; as described in [41], this can be resolved by additional protocol steps. We also remark that it is possible to develop other protocols in which the TTP does not have direct access to the IP and SW; interested readers are again referred to [41].

There is a second, important security use of the power-up states of SRAM cells that should not go unmentioned: Holcomb et al. show that those SRAM cells whose power-up states are unstable (i.e., those whose power-up states flip randomly between zero and one from power-up to power-up) can be used as a hardware-internal random number generator [49, 50].

Finally, a number of alternative hardware implementations have been proposed, using other PUF types that are similar to SRAM PUFs: For example so-called Butterfly PUFs [63] or Buskeeper PUFs [139]. Also the use of diodes has been proposed [52, 105].

4.3 *Remote Identification by Light Scattering in Random Media*

Our last example is an identification scheme suggested by Pappu et al. [88, 89] in 2001/2002, which rests on optical interference phenomena. At the heart of their method is a transparent, cuboid-shaped plastic platelet of size 1 cm $\times$ 1 cm $\times$ 2.5 mm, in which a large number of micrometer-sized glass spheres have been distributed randomly during the production process. The varying sizes, shapes and positions of the spheres induce strong disorder in the platelet, making it practically infeasible to build two specimen which are exactly the same. The platelet is "*unclonable*" by use of current technology, similar to the examples that we discussed earlier.

When a laser beam is directed at the platelet, the light is scattered multiple times in transition, creating a so-called "*speckle pattern*", i.e., an interference pattern of dark and bright regions. This pattern can be recorded conveniently by a CCD camera placed behind the platelet. Asides from the relative positioning of the platelet and the camera, which we imagine as fixed and do not consider further here, this speckle pattern sensitively depends on:

(a) the random positions, sizes and shapes of the spheres (i.e., on the disorder inside the token), and
(b) on the angle α and point $\mathbf{r}$ of incidence of the laser beam.

The latter can be varied, with each new pair of parameters $(\mathbf{r}, \alpha)$ leading to new patterns. Leaving aside measurement noise, or assuming perfect error correction, the input-output behavior of the token as a function f that maps measurement parameters $(\mathbf{r}, \alpha)$ into speckle patterns $f(\mathbf{r}, \alpha)$. The situation is depicted in Fig. 2.

The function f has a number of interesting properties. First, f possesses a very large number of input-output pairs $((\mathbf{r}, \alpha), f(\mathbf{r}, \alpha))$. In the parlance of the field, these are also called "*challenge-response pairs (CRPs)*", with $C = (\mathbf{r}, \alpha)$ being

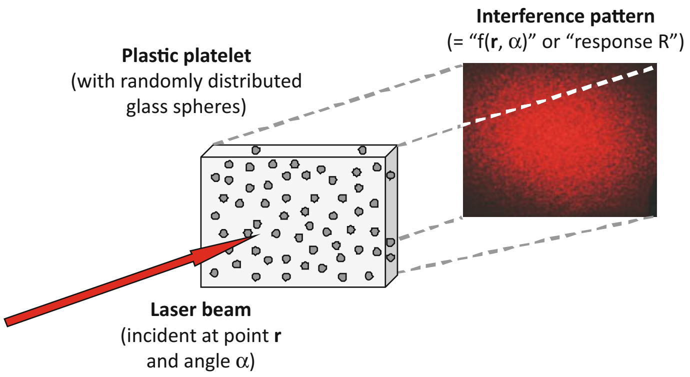


Fig. 2 Illustration of Pappu's optical, interference-based physical one-way function [88, 89]

the challenge, and $R = f(\mathbf{r}, \alpha)$ being the response. Pappu et al. estimate that the above platelet size allows around $2.37 \cdot 10^{10}$ challenges/inputs which lead to computationally independent speckle patterns as responses/outputs. If an adversary has got access to the platelet merely for a limited time period on the order of days or weeks, he will find himself unable to measure all possible input-output pairs and to complete characterize the function f. Secondly, an adversary knowing only a subset of all challenge-response pairs will be unable to numerically predict the speckle pattern to a new, unknown input $\mathbf{r}, \alpha$ without making a physical measurement on the token. The main reason is that the input-output behavior of the object is too laborious to simulate on a computer. As analyzed by Pappu et al. [89], in the worst case every cubic subunit of the platelet whose side length is around the wavelength λ of the incoming laserlight would play a role in the scattering process. For a cube with side length 1 cm, this leads to one Terabit of relevant subunits whose interaction would need to be considered in a simulation, making the latter practically infeasible. Similar considerations, thirdly, hold for the non-invertibility of f: Given a speckle pattern, it is practically impossible to determine which challenge $C = (\mathbf{r}, \alpha)$ created this speckle pattern, even if one has access to the token. This non-invertibility property of f originally inspired the name "*physical one-way function*" or "*POWF*" [88, 89] for the above structure; today, it is often refereed to as "*optical PUF*".

How can these properties of f, and of POWFs in general, be exploited in cryptography and security? How can we make use of their unclonability and unpredictability? Perhaps their best known security application are identification protocols, for example in a bank card scenario. In the following protocol, k is the security parameter, and l is the number of envisaged executions of the identification phase.

Protocol 3: BANK CARD IDENTIFICATION WITH LIGHT SCATTERING TOKENS

Set-Up and Security Assumptions:

1. The bank can securely store secret data on some server.
2. Each bank terminal is connected to the bank server by a non-confidential, but authenticated channel.
3. The bank can fabricate light scattering platelets or has access to a trusted manufacturer.

Initialization Phase:

1. The bank fabricates a light scattering platelet or obtains such a platelet from a trusted manufacturer. It attaches it as token to a bank card, which bears the customer identification number ID.
2. The bank chooses at random $k \cdot l$ parameters $\mathbf{r}_i, \alpha_i$, and applies a laser beam at position $\mathbf{r}_i$ and under angle α_i to the token. It measures the resulting speckle patterns, and derives from the raw data the error-corrected responses $R_i = f(\mathbf{r}_i, \alpha_i)$, for example by applying image transformation or error correction.
3. The bank stores the list $\mathcal{L}_{\mathsf{ID}} = (\mathbf{r}_1, \alpha_1, R_1), \ldots, (\mathbf{r}_{k \cdot l}, \alpha_{k \cdot l}, R_{k \cdot l})$ together with the identification number ID of the card on its server.

4. The bank card is released to the field.

Identification Phase (can be executed maximally l times):

1. When the card is inserted into a terminal, the terminal reads the ID from the card and sends ID to the server.
2. The server looks up the list $\mathcal{L}_{\mathsf{ID}}$. It chooses the first k entries $(\mathbf{r}_1, \alpha_1, R_1), \ldots, (\mathbf{r}_k, \alpha_k, R_k)$ from the list, and sends the parameters $(\mathbf{r}_1, \alpha_1), \ldots, (\mathbf{r}_k, \alpha_k)$ to the terminal.
3. The terminal applies laser beams with the incidence coordinates and angles given by $(\mathbf{r}_1, \alpha_1), \ldots, (\mathbf{r}_k, \alpha_k)$ to the token, and measures the corresponding speckle patterns. It derives the responses $R'_1 = f(\mathbf{r}_1, \alpha_1), \ldots, R'_k = f(\mathbf{r}_k, \alpha_k)$ from the raw data by applying the same image transformations or error correction as the bank in the set-up phase.
4. The terminal returns $R'_1, \ldots, R'_k$ to the server.
5. The server compares the responses $R_1, \ldots, R_k$ and $R'_1, \ldots, R'_k$. If they match better than given error threshold, the server sends an "OK!" message to the terminal. Otherwise, it sends an abort message.
6. The first k entries are erased from the list $\mathcal{L}_{\mathsf{ID}}$.

4.3.1 Security Discussion

A meaningful discussion requires us to first fix the underlying attack model. It is reasonable to assume that an attacker will be able to physically access the plastic platelet several times between different executions of the identification phase: He could set up faked terminals, or gain possession of the bank card when the customer employs it on other occasions, for example for paying in shops or restaurants, etc. Furthermore, we should suppose that the attacker can eavesdrop the binary communication in the identification protocol and learn the used CRPs $(C_1, R_1), \ldots, (C_k, R_k)$. Under this relatively strong attack model, the security of the scheme is nevertheless upheld by the above properties of the physical one-way function f and of the token. An adversary will be (1) unable to clone the token physically, and (2) cannot predict the unknown input-output-pairs (or CRPs) numerically, even if he knows a large number of other CRPs. This renders him unable to complete the identification protocol successfully without actual possession of the real token, guaranteeing the security of the above identification method. Interestingly, the scheme does not utilize the one-way property of f, but only the features of unclonability and unpredictability.

4.3.2 Potential Advantages and Drawbacks

Compared to standard identification schemes, Pappu et al.'s method exhibits a few notable advantages. First and foremost, no secret digital keys need to be stored on the bank card. Assuming that the token is too complex to simulate and rebuild, there is indeed no security-critical information at all present on the card whose disclosure would break the security of the system. Even if the adversary knew all positions of the scatterers and all irregularities of the structure, he still could not rebuild or simulate it, since both would be practically infeasible. We can allow him to possess any information in the scattering object without endangering the security of the scheme! This feature is in sharp contrast to any classical techniques, which necessitate that at least some information on the card remains secret. It is also in contrast to some PUF-based techniques, for example the SRAM PUFs presented in Sect. 4.2, where a disclosure of the SRAM power-up states to the adversary breaks the security of the system. Secondly, no potentially laborious numerical identification schemes need to be executed on the card in Pappu et al.'s scheme. The card does not even need to carry integrated circuitry, making it extremely cost effective, at least on the card side. Furthermore, it seems to potentially realize improved security against side channel attacks, which circuit implementations of classical identification schemes would be faced with. One last advantage is that the scheme enables remote identification (to the bank headquarters in our case) without circuitry on the card.

The scheme's main drawback is perhaps its mediocre practicality: The apparatus for measuring the speckle pattern (i.e., the response of the optical PUF) is expensive, bulky and potentially error prone. The extra costs for the measurement apparatus can partly eat up the savings from avoiding electrical circuitry on the card.

4.3.3 Variants

There are a number of variants of the above scheme. Firstly, other hardware systems than the described optical PUF can be employed. Any other so-called Strong PUFs [109] can be used, for example Arbiter PUFs and variants [36, 77, 144], the Bistable Ring PUF [18, 19], PUFs based on non-linear current mirrors [62], or PUFs based on non-linear voltage transfer characteristics [157], as long as they are secure against modeling [35, 66, 78, 87, 103, 106, 111] and other attacks, for example side channels [30, 74, 81, 82, 113]. Also secure integrated optical PUFs would be an option, preferably with non-linear scattering materials (compare [110]). Finally, the hardware of optical PUFs can be used for a number of other, more advanced cryptographic protocols, including key exchange [13, 97, 154], bit commitment [13, 86, 88, 100, 101], or oblivious transfer [13, 86, 95].

5 Advantages of Disorder-Based Security Hardware

Let us condense and summarize the advantages of disorder-based security hardware in this section. We thereby take a pure hardware-centered perspective, ignoring some of the specific cryptographic advantages.

Better Protection or Even Avoidance of Keys

One of the most important upsides of disorder-based techniques lies in their relation to cryptographic keys. All techniques of the last Sect. 4 avoid the presence of "*classical secret keys*" in vulnerable hardware, i.e., the presence of digital keys that are stored permanently in NVM. This has been described in all detail throughout Sect. 4.

Some of the presented approaches go even one step further, however, which can be seen most easily if we generalize the notion of a classical secret key. Let us call a "*security-critical information*" (*SCI*) any information that is present in a piece of hardware at least at one point in time, and whose disclosure to the adversary breaks the security of the system. One can then ask: Do the hardware systems of Sect. 4 contain any *SCI* in the above sense? Please note that this question goes far beyond our earlier point of avoiding the *permanent* presence of *digital* keys.

The answer differs for the three systems of Sect. 4. To start with, the paper structure of system Sect. 4.1 does not contain any security-critical information at all, since the adversary would be unable to refabricate the complex paper structure, even if he knew it atom by atom. Something similar holds for the optical PUF of Sect. 4.3: It could not be cloned, and its output could not be simulated for complexity reasons, even if the entire structure would be known to the adversary in arbitrary detail. On the other hand, the SRAM PUFs of Sect. 4.2 lead to systems that do contain SCI: The power-up states of the SRAM cells constitute SCI; and so does the internal key obtained from the power-up states after error correction. In this sense, the SRAM PUFs differ from optical PUFs, or from the paper based COAs of Sect. 4.1.

We would like to stress that this distinction is not just academic, but has a direct practical relevance. For example, the presence of SCI eventually enables the invasive attacks on SRAM PUFs that recently have attracted considerable attention [84]. Furthermore, the delays in Arbiter PUFs also represent a form of SCI, a fact that eventually allows modeling attacks on this type of structure [106]. In essence, the presence of SCI in a hardware system necessarily creates unwanted attack points, and well-versed adversaries will in the end exploit these. This makes it preferable to construct disorder-based systems without any SCI, and disorder-based methods offer this perhaps stunning possibility. We would like to encourage readers to pay increasing attention to the above distinction, and to categorize disorder-based security approaches with respect to the feature of "*being free of any SCI*" whenever possible.

Replacing Standard Mathematical Assumptions by Other Hypotheses

A noteworthy theoretical aspect of disorder-based schemes is that they avoid mathematical assumptions in cryptographic and security schemes. Take the identification scheme of Sect. 4.3 as an example: Its security is not based on the same standard mathematical and number-theoretic assumptions as traditional identification methods. It does rest on computational assumptions, too, namely the fact that the optical response cannot be simulated efficiently, and also depends on other, physical assumptions, like the unclonability of the optical PUF. But these hypotheses notably differ from the standard mathematical assumptions like factoring or discrete logarithm. In this sense, PUF-based cryptography could be seen in the context of other non-standard approaches, such as quantum cryptography [8], noise-based crypto [25], or the bounded storage model [5, 79]. It is also in alignment with "*post-quantum cryptography*" [9], since the underlying computational problems are not known to be attackable by quantum computers.

This theoretical aspect also has practical consequences: Identification schemes based on so-called "*Strong PUFs*" [98, 104, 109], like the method of Sect. 4.3, do not require the implementation of standard identification schemes in hardware. Such implementations have always been a potential target for attacks in the past, including side channels. This target can potentially be removed by use of Strong PUF based identification schemes. On the other hand, it should not go unmentioned that side channel attacks on certain *electrical* PUFs have been reported recently, even though this subarea of PUF research is just about to develop [30, 74, 113].

Security Hardware Without NVM or Even Without ICs

From a practicality perspective, the most important upside of the techniques of Sect. 4 is that they enable security functionalities in hardware without NVM, and partly even in hardware without integrated circuits (ICs). The use of SRAM cells on FPGAs without NVM (Sect. 4.2) is one known example for the former, while the exploitation of surface irregularities or optical PUFs (Sects. 4.1 and 4.3) are examples for the latter. Both can be decisive practicality and cost factors, as they bring security to systems where otherwise elaborate and dedicated security measures would be impossible. Recall in this context that adding non-volatile keys to hardware systems without NVM requires significant additional production steps and costs, disallowing it in certain commercial settings.

6 General Overview and Historic Perspective

This chapter would not be complete without a general overview and historic perspective of the area. Our discussion reveals that the field has older and broader roots than usually acknowledged. It also shows the different branches of research in disorder-based security: While physical unclonable functions are clearly the central and dominant subfield, there are also other noteworthy subareas, which should be distinguished from PUFs for historic or scientific reasons. Providing such a basic distinction will be useful in inspiring and guiding future research, we believe.

Origins of the Field

It is non-trivial to trace back the field to its exact origins. To the knowledge of the author, the first public source utilizing random, uncontrollable manufacturing variations in a security context is a US-patent with priority date 1968 by Lindstrom and Schullstrom of Saab AB, Sweden [68]. It suggests that randomly, non-uniformly distributed magnetic materials could be employed for individualizing and securing "*identification documents like driver's licences and credit cards*". It further proposes that concealed, internal layers of such materials might protect sensitive regions of identification documents, for example the picture of the card holder, against alteration, and could detect manipulation of these regions. The latter foreshadows a security feature that today is called tamper-sensitivity. The inventors also suggest that electrical or optical materials could be used to the same end.

It has also been reported that in the 1970s, Bauder and Simmons of Sandia National Laboratories, USA, exploited the optical behavior of physically disorder media for security purposes [17, 59, 83] (Kirovski, 2008, Personal communication, Dagstuhl). Their goal was to conduct secure weapons inspection during the cold war era. To this end, they reportedly spray-painted epoxy onto nuclear warheads, shed light at it from a certain angle, and recorded pictures of the resulting optical patterns [17, 59, 83] (Kirovski, 2008, Personal communication, Dagstuhl). These images could later be used to re-identify each single warhead in a forgery-proof manner [17, 59, 83] (Kirovski, 2008, Personal communication, Dagstuhl).

Perhaps the first to combine modern public-key methods with physical disorder was Goldman of Light Signatures Inc., USA. In a patent writing with priority date 1980, he details the use of paper irregularities in connection with digital signatures for certifying documents [40]. Light Signatures commercialized this technique in order to authenticate stock certificates in the mid 1980s, but their activities were apparently not profitable and abandoned in 1988 [119]. Presumably independently of Goldman, Bauder [reportedly together with Simmons [17], Kirovski (2008, Personal communication, Dagstuhl)] suggested a similar concept at Sandia National Laboratories for the protection of banknotes, also combing unique paper structures with digital signatures [59]. A Sandia-internal source that is multiply quoted in this context is by Bauder [6], dating from 1983.[7]

[7]However, copies of this paper seem unavailable to a broad public. The author of this chapter has been unsuccessful in gaining access despite considerable efforts, including multiple e-mail requests to the Sandia National Laboratories. Other researchers made partly similar experiences (Kirovski, 2008, Personal communication, Dagstuhl).

With some right, the three above, independent research groups could be seen as early forefathers of disorder-based security and also of physical unclonable functions. This would imply that the field has older roots than sometimes acknowledged.

First Presence at Scientific Conferences

The groundbreaking ideas of Lindstrom and Schullstrom, Goldman, and Bauder and Simmons, seemingly were not discussed much in public scientific conferences or journals until the 1990s. Perhaps the earliest scientific paper that points in the relevant direction is by Simmons, dating from 1991 [138]. Independently and a few years later, a number of publications by van Renesse treat similar ideas, focusing on optical product protection systems [155, 156]. Suggestions based on magnetic materials, which are in principle related to the early patent of Lindstrom and Schullstrom, have been made independently by Chu et al. [20] and Vaidya [151] at scientific venues in 1995.

In 1998, Haist et al. [43] also discuss paper and optical probing for product protection, making explicit use of digital signatures in a similar fashion as Goldman. A closely related scientific publication is by Smith et al. [141] from 1999. It uses paper irregularities with digital signatures in order to create unforgeable postal stamps.

In 2000, Lofstrom et al. [69] for the first time suggest the variations in standard integrated circuit components for security purposes, exploiting the random threshold mismatches in transistors to identify individual circuits (compare Sect. 4.2). Their paper could be seen as a direct precursor of the modern PUF era, foreshadowing so-called intrinsic PUFs like SRAM PUFs and Butterfly PUFs (without explicit use of the term “PUF”, though).

DNA-Based Steganography

Before we eventually turn to PUFs, let us quickly mention another independent research avenue. In 1999, the randomness in complex mixtures of DNA strands was suggested for use in security and steganography by Clelland et al. in Nature magazine [22]. If a secret message or other critical information is encoded in DNA strands, and if these strands are mixed with a huge number of other, random strands, an adversary would find it practically impossible to identify and isolate the “secret” strands. He would be faced with the proverbial search for the needle in the haystack. The fact that complex DNA mixtures can be generated by simple means adds to the value of this method [22]. In follow-up work, DNA-based public and private key cryptography has been discussed, for example, in [39, 64]. The approach of Clelland et al. has even led to commercially available products [120].

DNA-based security might appear off topic and seems generally less known within the PUF community. Still, it has established its own research strand, with hundreds of citations, presence at DNA-related scientific venues, and a certain level of commercial activities [120–122]. Furthermore, it historically seems the first approach that ever exploited truly nanoscale phenomena for security. This foreshadows a recently emerging trend towards nano-security in the PUF area [52, 93, 107, 108].

Physical Unclonable Functions (PUFs)

Despite all above contributions, it seems fair to say that the interest of the broader security community was not sparked until 2001/02, when a few seminal works were published at major scientific venues: Firstly, Pappu [88] in 2001, and Pappu et al. [89] in Science magazine in 2002, presented the idea of so-called "*physical one-way functions*" or "*POWFs*". Secondly, Gassend et al. published the concept of silicon, circuit-based "*physical random functions*" at ACM CCS 2002 [36], and of "*controlled physical random functions*" at ACSAC 2002 [37]. The latter two papers also use the term "*physical unclonable function*" or "*PUF*" for the first time, which today is frequently employed as a synonym for the entire research area.

Compared to earlier works, said four publications make a number of central innovations. Among other things, they are the first to link disordered, unclonable media to more established cryptographic and security concepts, including one-way functions or pseudo-random functions. Moreover, they use disordered systems with a very large number of possible inputs, whose input/output behavior could be regarded as a complex "*physical function*" [4]. The mathematical properties of this function, such as unpredictability or one-wayness, can then be explicitly exploited in cryptographic protocols. This new view of the field, and the links it establishes to known concepts, helped attracting the interest of the broader security community, and contributed to spreading the new approach quickly.

Other seminal PUF works in the early period from 2002 to 2007 include (but are not limited to): The AEGIS security architecture by Suh et al. [145] from 2003; first information-theoretic analyses of PUFs by Tuyls et al. in 2004/2005 [148, 149]; the security use of laser illuminated paper surfaces by Buchanan et al. in Nature in 2005 [14] (compare Sect. 4.1); and the usage of disordered electrical structures as tamper sensitive coatings by Tuyls et al. [150] in 2006. A further groundbreaking idea was the use of the individual, but repeatable power-up states of SRAM cells as secret key source. This concept is particularly useful in hardware that does not carry non-volatile memory cells. It was independently put forward by Holcomb et al. [49] and Guajardo et al. [41] in 2007.

Certificates of Authenticity (COAs)

Starting a few years later than PUFs, a parallel and independent strand of works helped popularizing the idea of disorder-based security. The strand roots quite directly in the original ideas of Lindstrom and Schullstrom, Goldman, and Bauder and Simmons, using very similar techniques: It combines the unique and unclonable features of disordered media with digital signatures to form so-called "*certificates of authenticity*" or "*COAs*" for objects of value. Example works include COA-specific error correction [57, 58] in 2004, optical COAs [17] (which pick up the early ideas of Bauder and Simmons [6]) in 2005, and radiowave based COAs [29] in 2007. Also work on unique optical fingerprints of compact discs by DeJean et al. [158] and Hammouri et al. [44] in 2009 could be associated with this strand. A good summary of the subarea is given by Kirovski in [59]. Most COA papers are somewhat demarked from PUFs in terms of nomenclature and scientific content. Furthermore, the COA-idea arguably dates back earlier than PUFs, being present in its full-fledged form in combination with digital signatures already in the 1980s [6, 40]. We thus found it appropriate to devote a separate paragraph to it. At the same time, we stress that the current focus of the community appears to be on PUFs, both regarding research activities, quotation numbers, and nomenclature.

Status Quo

From 2008 onwards, a rapidly growing activity on disorder-based security and PUFs takes place. It is mostly, but not exclusively focused on PUFs, and often regards the two works by Pappu et al. [89] and Gassend et al. [36] as root publications of the field. Listing all published works of the last 6 years is beyond the intention and scope of this section; we refer the interested reader to recent survey articles [71, 98, 109] or PUF bibliographies [118] to this end.

Instead, we will list several exemplary facts that testify the rapid establishment of the area. To start with, according to Google scholar, the two root articles of Pappu et al. [89] and Gassend et al. [36] have been quoted many hundred times to date, with increasing citation figures almost every year. Since 2008, papers on PUFs and related topics have been published at CHES [10, 12, 44, 48, 54, 55, 61, 70, 70, 72, 73, 85, 100, 146, 152, 163, 167], EUROCRYPT [86], ASIACRYPT [3, 27], CRYPTO [13], ACM CCS [106, 136], IEEE S&P [4, 102], IEEE T-IFS [23, 76, 111], ACM TISSEC [38], and the Journal of Cryptology [75], i.e., in all top publication channels of the general cryptography and security community. Since 2010, the two large hardware security conferences CHES and HOST continuously had one or even two dedicated PUF sessions each year (see [115, 116]). DATE, one of the two largest design automation conferences, in 2014 offered both a standard technical session on PUFs [123], a hot topic session on PUFs [112, 124], a related tutorial on counterfeiting ICs [125], and a record of overall eleven (!) PUF papers at one single conference [130]. This extraordinary accumulation is not just one singular event, but the culmination of a longer development: DATE already from 2010 to 2013 [19, 28, 60, 114, 130, 142, 153, 159, 164, 166], and its counterpart DAC from 2009 onwards [16, 34, 46, 47, 80, 91, 131, 165, 168], continuously hosted at least one PUF-paper every year, illustrating that PUFs have long spread from their original field of security into neighboring areas like circuit design. Also the first coursebooks on PUFs have appeared lately [11]. Even on the commercial side, PUFs achieved some recent breakthroughs, appearing in the product lines of major companies like NXP [126, 127] and Microsemi [128, 129].

It therefore seems justified to say that almost 15 years after its popularization in scientific circles [36, 37, 69, 88, 89], and around 45 years since its very first presence in patent writings [68], the field has developed into a central subarea of hardware security, and currently shows no signs of slowing down in its rapid progress. Particularly strong research activity thereby have appeared and still appear in the fields of PUF hardware implementations; error correction; PUF protocols; and PUF attacks. Again, we refer the interested readers to recent surveys for a detailed treatment of the very latest PUF literature [98].

Future Research

It is always difficult to predict exact future developments in a field as complex as PUFs. But in our opinion, there are good reasons to believe that the most promising and active subareas in the upcoming years will likely include the following:

- *Formalization and Classification*: A solid formalization of the area, including formal definitions and security proofs, seems vital for a sound long-term development of the field. It should include a classification and specification of the different primitives, their basic applications, and the exact security features required by a given application. Such specification will be essential not only for a sound long-term development of the field, but also for an efficient communication within multi-disciplinary PUF research teams, ranging from programmers over application designers to electrical engineers.
- *Attacks and Countermeasures*: The large number of recent, successful attacks has shown that PUFs are no magic toolbox that can create security from nothing. Extending existing attack techniques, and possibly creating entirely new ones, will likely constitute a very active field of research in future years. The same holds for the development and implementation of countermeasures. This process will partly be comparable to the consolidation that classical security hardware has undergone already several years ago.
- *Applications of Nanotechnology*: Nanotechnology and security are two of the most active and prosperous subdisciplines in the sciences in the last decades. Disorder-based security lies exactly at their intersection. It seems very likely that a merger between the two fields will create new research opportunities, partly

exceeding the current topics in the area. It will allow new primitives and security features and new, advantageous implementations.

- *New Uses of Physical Disorder in Security, and new Disorder-Based Primitives*: Besides PUFs, COAs, and related existing methods, there seems to be potential for new, currently unseen uses of physical disorder in security and new disorder-based primitives. Any new discovery will spark and guide new research, keeping the field prospering.
- *Improved or New Implementations*: PUFs and all other abstract concepts within disorder-based security can only shine if suitable hardware realizations of these concepts are developed. Strong and continuous research potential therefore lies in improving existing implementations of disorder-based primitives, or in the development of yet new ones. Such optimization might be carried out with hindsight to costs, stability, area and size, power consumption, security, or yet other aspects.
- *Commercialization*: One last subfield where a predictably strong activity will occur lies in the area of commercialization. What are the exact use cases of PUFs in commercial contexts? How can the scientific developments of recent years be applied and utilized in concrete commercial scenarios? How can PUFs be integrated into commercial system?

7 Summary and Outlook

In this chapter, we surveyed a recently emerging subfield of hardware security that could be called "*disorder-based security*" or also "*nano-security*". It exploits the small-scale, random physical disorder that is present in essentially all solid state systems. The roots of the field reach back surprisingly far, with first appearances in patent writings in the late 1960s. Today, its most active subfield are by far physical unclonable functions (PUFs); two other, but considerably smaller subareas, which might be distinguished for historic and also scientific reasons, are DNA-based cryptography and so-called "*certificates of authenticity*" (COAs).

The main motivation for disorder-based hardware techniques are practicality/cost and security advantages. Both relate to the way we currently treat secret keys in security hardware: Usually, the keys are stored in non-volatile memory cells (NVM) or comparable structures, and are processed by mathematical crypto-algorithms. Disorder-based methods, for example PUFs and COAs, offer several potential improvements in this situation.

Firstly, they allow keys in hardware without NVMs, potentially cutting on costs. Recall that not all hardware systems possess NVMs, one much discussed example being certain types of FPGAs. Using disorder-based methods, keys can potentially be derived from the hardware-internal physical disorder instead of dedicated NVM. One prime example are SRAM cells (see Sect. 4.2): The cells show individual power-up states due to small manufacturing variations, and can hence be used as key source. A second example are optical PUFs (see Sect. 4.3), which even allow remote identification without electrical circuitry in the hardware itself.

Secondly, disorder-based methods avoid the long-term presence of keys in digital form in the hardware. Again, this leads to potential security advantages: Keys derived from SRAM cells, for example, are present in the hardware system in digital form only for very short times frames; they are only derived whenever needed by a cryptographic algorithm. This can make attacks more difficult: As long as the system is powered off, it is more difficult or even impossible for adversaries to obtain the keys. Going yet one step further, systems like optical PUFs or paper-based COAs do not contain any type of keys or security critical information at all; security systems built on them remain secure even if an adversary knows their structure atom by atom. This unusual and unexpected feature of these systems appears to be a major asset in comparison with standard approaches.

Thirdly, certain disorder-based methods can replace the use of mathematical cryptographic schemes. One example is once more the optical PUF based identification scheme of Sect. 4.3, which makes the employment of classical, mathematical algorithms for identification purposes obsolete. This can potentially increase security levels, too: As a first aspect, the resulting schemes no longer rest on the unproven standard mathematical assumptions like the intractability of factoring or of computing discrete logarithms (albeit on other unproven assumptions). Interestingly, this is in alignment with the area of "*post-quantum cryptography*" [9] and its recent efforts. As a second aspect, the concrete implementation of mathematical algorithms in hardware has always been a potential target for attacks in the past, including side channel techniques. This target is partly removed if no such mathematical algorithms are necessary in identification schemes. On the other hand, it should not go unnoticed that also side channel attacks on certain *electrical* PUFs have been reported recently, even though this area is just about to develop [30, 74, 113].

We believe that the field will continue to rapidly expand and flourish in the foreseeable future. Current publication activity is extraordinarily high, and the presence at scientific venues has been steadily increasing. It seems important to mention that besides a very large number of scientific innovations, also some first commercial breakthroughs could be achieved recently [126–129]. Regarding future activities, six promising subareas lie in (1) formalization and classification, (2) attacks and countermeasures, (3) the merger of nanotechnology and security, (4) new uses of physical disorder in security, including new disorder-based primitives, (5) improved or new implementations, and finally (6) the integration in commercial systems. The fact that these subareas span from theoretical computer science over electrical engineering to nanophysics gives the field an unusually broad focus and attractivity.

References

1. Adams, D.: The Hitchhiker's Guide to the Galaxy. Pan Books, London (1979)
2. Anderson, R.J.: Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd edn. Wiley, New York (2008)

3. Armknecht, F., Maes, R., Sadeghi, A.-R., Sunar, B., Tuyls, P.: Memory leakage-resilient encryption based on physically unclonable functions. In: Proceedings of ASIACRYPT 2009, pp. 685–702 (2009)
4. Armknecht, F., Maes, R., Sadeghi, A.-R., Standaert, F.-X., Wachsmann, C.: A formal foundation for the security features of physical functions. In: IEEE Symposium on Security and Privacy 2011, pp. 397–412 (2011)
5. Aumann, Y., Ding, Y.Z., Rabin, M.O.: Everlasting security in the bounded storage model. IEEE Trans. Inf. Theory **48**(6), 1668–1680 (2002)
6. Bauder, D.W.: An anti-counterfeiting concept for currency systems. Technical Report, PTK-11990. Sandia National Labs, Albuquerque, NM (1983)
7. Beckmann, N., Potkonjak, M.: Hardware-based public-key cryptography with public physically unclonable functions. In: Information Hiding 2009, pp. 206–220. Springer, Heidelberg (2009)
8. Bennett, C.H., Brassard, G.: Quantum cryptography: public key distribution and coin tossing. IEEE Int. Conf. Comput. Syst. Signal Process. **175**(150), 8 (1984)
9. Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.): Post-Quantum Cryptography. Springer, Berlin (2009) [ISBN 978-3-540-88701-0]
10. Bhargava, M., Mai, K.: A high reliability PUF using hot carrier injection based response reinforcement. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013, pp. 90–106. Springer, Heidelberg (2013)
11. Böhm, C., Hofer, M.: Physical Unclonable Functions in Theory and Practice. Springer, Berlin (2013) [ISBN 978-1-4614-5040-5]
12. Bösch, C., Guajardo, J., Sadeghi, A.-R., Shokrollahi, J., Tuyls, P.: Efficient helper data key extractor on FPGAs. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008, pp. 181–197. Springer, Heidelberg (2008)
13. Bruzska, C., Fischlin, M., Schröder, H., Katzenbeisser, S.: Physical unclonable functions in the universal composition framework. In: CRYPTO 2011, pp. 51–70 (2011)
14. Buchanan, J.D.R., Cowburn, R., Jausovec, A., Petit, D., Seem, P., Xiong, G., Atkinson, D., Fenton, K., Allwood, D., Bryan, M.: Fingerprinting documents and packaging. Nature **436**(7050), 475 (2005)
15. Buchmann, J., May, A., Vollmer, U.: Perspectives for cryptographic long-term security. Commun. ACM **49**(9), 50–55 (2006)
16. Chakraborty, R., Lamech, C., Acharyya, D., Plusquellic, J.: A transmission gate physical unclonable function and on-chip voltage-to-digital conversion technique. In: Design Automation Conference (DAC), paper no. 59, 10 p. (2013)
17. Chen, Y., Mihcak, M.K., Kirovski, D.: Certifying authenticity via fiber-infused paper. SIGecom Exch. **5**(3), 29–37 (2005)
18. Chen, Q., Csaba, G., Lugli, P., Schlichtmann, U., Rührmair, U.: The Bistable ring PUF: a new architecture for strong physical unclonable functions. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST 2011), pp. 134–141 (2011)
19. Chen, Q., Csaba, G., Lugli, P., Schlichtmann, U., Rührmair, U.: Characterization of the Bistable Ring PUF. In: Design, Automation and Test in Europe (DATE), pp. 1459–1462 (2012)
20. Chu, M.C., Cheng, L.L., Cheng, L.M.: A novel magnetic card protection system. European Convention on Security and Detection, pp. 207–211 (1995)
21. Clarkson, W., Weyrich, T., Finkelstein, A., Heninger, N., Halderman, J.A., Felten, E.W.: Fingerprinting blank paper using commodity scanners. In: IEEE Symposium on Security and Privacy 2009, pp. 301–314 (2009)
22. Clelland, C.T., Risca, V., Bancroft, C.: Hiding messages in DNA microdots. Nature **399**(6736), 533–534 (1999)
23. Cobb, W.E., Laspe, E.D., Baldwin, R.O., Temple, M.A., Kim, Y.C.: Intrinsic physical-layer authentication of integrated circuits. IEEE Trans. Inf. Forensics Secur. **7**(1), 14–24 (2012)
24. Cox, I.J., Miller, M.L., Bloon, J.A., Fridrich, J., Kalker, T.: Digital Watermarking and Steganography. Morgan Kaufmann, Elsevier (2008)

25. Crepeau, C.: Efficient cryptographic protocols based on noisy channels. In: EUROCRYPT 1997, pp. 306–317 (1997)
26. Csaba, G., Ju, X., Ma, Z., Chen, Q., Porod, W., Schmidhuber, J., Schlichtmann, U., Lugli, P., Rührmair, U.: Application of mismatched cellular nonlinear networks for physical cryptography. In: IEEE CNNA 2010, pp. 1–6 (2010)
27. Damgard, I., Scafuro, A.: Unconditionally Secure and Universally Composable Commitments from Physical Assumptions. In: ASIACRYPT 2013, pp. 100–119 (2013)
28. Das, A., Kocabas, Ü., Sadeghi, A.-R., Verbauwhede, I.: PUF-based secure test wrapper design for cryptographic SoC testing. In: Design, Automation and Test in Europe (DATE), pp. 866–869 (2011)
29. DeJean, G., Kirovski, D.: RF-DNA: Radio-frequency certificates of authenticity. In: Proceedings of CHES 2007, pp. 346–363 (2007)
30. Delvaux, J., Verbauwhede, I.: Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 137–142 (2013)
31. Deutsch, D.: Quantum theory, the Church-Turing principle and the universal quantum computer. Proc. R. Soc. Lond. A Math. Phys. Sci. **400**(1818), 97–117 (1985)
32. Deutsch, D., Ekert, A., Luppachini, A.: Machines, Logic and Quantum Physics (1999). Downloaded from http://arxiv.org/abs/math.LO/9911150, August 2012 [arXiv:math/9911150v1]
33. Feynman, R.: Simulating physics with computers. Int. J. Theor. Phys. **21**(6/7), 467–488 (1982)
34. Forte, D., Srivastava, A.: On improving the uniqueness of silicon-based physically unclonable functions via optical proximity correction. In: Design Automation Conference (DAC), pp. 96–105 (2012)
35. Gassend, B.: Physical random functions. M.Sc. thesis, MIT (2003)
36. Gassend, B., Clarke, D.E., van Dijk, M., Devadas, S.: Silicon physical random functions. In: ACM Conference on Computer and Communications Security (ACM CCS), pp. 148–160 (2002)
37. Gassend, B., Clarke, D.E., van Dijk, M., Devadas, S.: Controlled physical random functions. In: Annual Computer Security Applications Conference (ACSAC), pp. 149–160 (2002)
38. Gassend, B., van Dijk, M., Clarke, D.E., Torlak, E., Devadas, S., Tuyls, P.: Controlled physical random functions and applications. ACM Trans. Inf. Syst. Secur. **10**(4), 3 (2008)
39. Gehani, A., LaBean, T., Reif, J.: DNA-based cryptography. Aspects of Molecular Computing, pp. 167–188. Springer, Berlin (2004)
40. Goldman, R.N.: Non-counterfeitable document system. US-Patent 4,423,415. Publication date: 1983. Priority date: 1980
41. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Proceedings of CHES 2007, pp. 63–80 (2007)
42. Gutmann, P.: Secure deletion of data from magnetic and solid-state memory. In: USENIX Security Symposium (1996)
43. Haist, T., Tiziani, H.J.: Optical detection of random features for high security applications. Opt. Commun. **147**(1), 173–179 (1998)
44. Hammouri, G., Dana, A., Sunar, B.: CDs have fingerprints too. In: Proceedings of CHES 2009, pp. 348–362 (2009)
45. Helfmeier, C., Boit, C., Nedospasov, D., Seifert, J.-P.: Cloning physically unclonable functions. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 1–6 (2013)
46. Helinski, R., Acharyya, D., Plusquellic, J.: A physical unclonable function defined using power distribution system equivalent resistance variations. In: Design Automation Conference (DAC), pp. 676–681 (2009)
47. Helinski, R., Acharyya, D., Plusquellic, J.: Quality metric evaluation of a physical unclonable function derived from an IC's power distribution system. In: Design Automation Conference (DAC), pp. 240–243 (2010)
48. Hofer, M., Böhm, C.: An alternative to error correction for SRAM-Like PUFs. In: Proceedings of CHES 2010, pp. 335–350 (2010)

49. Holcomb, D.E., Burleson, W.P., Fu, K.: Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In: Conference on RFID Security (2007)
50. Holcomb, D.E., Burleson, W.P., Fu, K.: Power-Up SRAM state as an identifying fingerprint and source of true random numbers. IEEE Trans. Comput. **58**(9), 1198–1210 (2009)
51. Image by Chipworks Inc. (www.chipworks.com): Oral permission for reproduction granted by D. James of Chipworks Inc. to the author on June 10, 2014. Downloaded from http://www.chipworks.com/components/com_wordpress/wp/wp-content/uploads/2013/08/A5-Processor-from-ipad-Mini-300x249.jpg (2014)
52. Jaeger, C., Algasinger, M., Rührmair, U., Csaba, G., Stutzmann, M.: Random pn-junctions for physical cryptography. Appl. Phys. Lett. **96**, 172103 (2010)
53. Kariakin, Y.: Authentication of articles. Patent writing, WO/1997/024699 (1995). Available from http://www.wipo.int/pctdb/en/wo.jsp?wo=1997024699
54. Katzenbeisser, S., Kocabas, Ü., van der Leest, V., Sadeghi, A.-R., Schrijen, G.-J., Schröder, H., Wachsmann, C.: Recyclable PUFs: logically reconfigurable PUFs. In: Proceedings of CHES 2011, pp. 374–389 (2011)
55. Katzenbeisser, S., Koçabas, Ü., Rozic, V., Sadeghi, A.-R., Verbauwhede, I., Wachsmann, C.: PUFs: myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon. In: Proceedings of CHES 2012, pp. 283–301 (2012)
56. Kerckhoffs, A.: La cryptographie militaire. J. Sci. Mil. **IX**, 5–38 (1883)
57. Kirovski, D.: Toward an automated verification of certificates of authenticity. In: ACM Electronic Commerce (EC), pp. 160–169 (2004)
58. Kirovski, D.: Point compression for certificates of authenticity. In: Data Compression Conference 2004, p. 545 (2004)
59. Kirovski, D.: Anti-counterfeiting: mixing the physical and the digital world. In: Sadeghi, A.-R., Naccache, D. (eds.) Towards Hardware-Intrinsic Security, pp. 223–233. Springer, Berlin (2010)
60. Koeberl, P., Kocabas, Ü., Sadeghi, A.-R.: Memristor PUFs: a new generation of memory-based physically unclonable functions. In: Design, Automation and Test in Europe (DATE), pp. 428–431 (2013)
61. Krishna, A.R., Narasimhan, S., Wang, X., Bhunia, S.: MECCA: a robust low-overhead PUF using embedded memory array. In: Proceedings of CHES 2011, pp. 407–420 (2011)
62. Kumar, R., Burleson, W.: On design of a highly secure PUF based on non-linear current mirrors. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST 2014), pp. 38–43 (2014)
63. Kumar, S.S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P.: The butterfly PUF: protecting IP on every FPGA. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 67–70 (2008)
64. Leier, A., Richter, C., Banzhaf, W., Rauhe H.: Cryptography with DNA binary strands. BioSystems **57**(1), 13–22 (2000)
65. Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. J. Cryptol. **14**(4), 255–293 (2001)
66. Lim, D.: Extracting secret keys from integrated circuits. M.Sc. thesis, MIT (2004)
67. Lim, D., Lee, J.W., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. IEEE Trans. VLSI Syst. **13**(10), 1200–1205 (2005)
68. Lindstrom, G., Schullstrom, G.: Verifiable identification document. US-Patent 3636318. Publication date: 1972. Priority date: 1968
69. Lofstrom, K., Daasch, W.R., Taylor, D.: IC identification circuit using device mismatch. In: International Solid-State Circuits Conference (ISSCC), pp. 372–373 (2000)
70. Maes, R.: An accurate probabilistic reliability model for silicon PUFs. In: Proceedings of CHES 2013, pp. 73–89 (2013)
71. Maes, R., Verbauwhede, I.: Physically unclonable functions: a study on the state of the art and future research directions. In: Sadeghi, A.-R., Naccache, D. (eds.) Towards Hardware-Intrinsic Security, pp. 3–37. Springer, Heidelberg (2010)

72. Maes, R., Tuyls, P., Verbauwhede, I.: Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In: Proceedings of CHES 2009, pp. 332–347 (2009)
73. Maes, R., Van Herrewege, A., Verbauwhede, I.: PUFKY: a fully functional PUF-based cryptographic key generator. In: Proceedings of CHES 2012, pp. 302–319 (2012)
74. Mahmoud, A., Rührmair, U., Majzoobi, M., Koushanfar, F.: Combined modeling and side channel attacks on strong PUFs. IACR Cryptology ePrint Archive, Report 2013/632 (2013)
75. Maiti, A., Schaumont, P.: Improved ring oscillator PUF: an FPGA-friendly secure primitive. J. Cryptol. **24**(2), 375–397 (2011)
76. Maiti, A., Kim, I., Schaumont, P.: A robust physical unclonable function with enhanced challenge-response set. IEEE Trans. Inf. Forensics Secur. **7**(1), 333–345 (2012)
77. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Lightweight secure PUFs. In: IC-CAD 2008, pp. 607–673 (2008)
78. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Testing techniques for hardware security. In: IEEE International Test Conference (ITC), pp. 1–10 (2008)
79. Maurer, U.: Conditionally-perfect secrecy and a provably-secure randomized cipher. J. Cryptol. **5**(1), 53–66 (1992)
80. Meguerdichian, S., Potkonjak, M.: Device aging-based physically unclonable functions. In: Design Automation Conference (DAC), pp. 288–289 (2011)
81. Merli, D., Schuster, D., Stumpf, F., Sigl, G.: Side-channel analysis of PUFs and fuzzy extractors. In: TRUST 2011, pp. 33–47 (2011)
82. Merli, D., Heyszl, J., Heinz, B., Schuster, D., Stumpf, F., Sigl, G.: Localized electromagnetic analysis of RO PUFs. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 19–24 (2013)
83. Mihcak, M.K.: Overview of recent content authentication research at MSR Crypto, Redmond. Available from https://www.yumpu.com/en/document/view/10835269/m-kivanc-mihcak-uvigo-tv, or from http://tv.uvigo.es/uploads/material/Video/91/Kivanc_Mihcak.pdf
84. Nedospasov, D., Seifert, J.-P., Helfmeier, C., Boit, C.: Invasive PUF analysis. In: Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 30–38 (2013)
85. Oren, Y., Sadeghi, A.-R., Wachsmann, C.: On the effectiveness of the Remanence decay side-channel to clone memory-based PUFs. In: Proceedings of CHES 2013, pp. 107–125 (2013)
86. Ostrovsky, R., Scafuro, A., Visconti, I., Wadia, A.: Universally composable secure computation with (malicious) physically uncloneable functions. In: EUROCRYPT 2013, pp. 702–718 (2013)
87. Öztürk, E., Hammouri, G., Sunar, B.: Towards robust low cost authentication for pervasive devices. In: IEEE PerCom 2008, pp. 170–178 (2008)
88. Pappu, R.: Physical one-way functions. Ph.D. thesis, MIT (2001)
89. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science **297**, 2026–2030 (2002)
90. Potkonjak, M.: Personal Communication (2011)
91. Potkonjak, M., Meguerdichian, S., Nahapetian, A., Wei, S.: Differential public physically unclonable functions: architecture and applications. In: Design Automation Conference (DAC), pp. 242–247 (2011)
92. Rivest, R.: Illegitimi non carborundum. Invited keynote talk, CRYPTO 2011. Downloaded from http://www.rsa.com/rsalabs/presentations/Riv11b.slides.pdf, August 2012
93. Rostami, M., Wendt, J.B., Potkonjak, M., Koushanfar, F.: Quo Vadis, PUF? Trends and challenges of emerging physical-disorder based security. In: Design, Automation and Test in Europe (DATE) (2014)
94. Rührmair, U.: SIMPL systems: on a public key variant of physical unclonable functions. IACR Cryptology ePrint Archive, Report 2009/255 (2009)
95. Rührmair, U.: Oblivious transfer based on physical unclonable functions (extended abstract). In: Proceedings of the 3rd International Conference on Trust and Trustworthy Computing (TRUST), pp. 430–440 (2010)
96. Rührmair, U.: SIMPL systems, or: can we build cryptographic hardware without secret key information? In: SOFSEM 2011. Lecture Notes in Computer Science. Springer, Heidelberg (2011)

97. Rührmair, U.: Physical turing machines and the formalization of physical cryptography. IACR Cryptology ePrint Archive, Report 2011/188 (2011)
98. Rührmair, U., Holcomb, D.E.: PUFs at a glance. In: Design, Automation and Test in Europe (DATE), pp. 1–6 (2014)
99. Rührmair, U., Holcomb, D.: PUFs at a glance. In: Design, Automation and Test in Europe (DATE 2014) (2014)
100. Rührmair, U., van Dijk, M.: Practical security analysis of PUF-based two-player protocols. In: Proceedings of CHES 20120, pp. 251–267 (2012)
101. Rührmair, U., van Dijk, M.: On the practical use of physical unclonable functions in oblivious transfer and bit commitment protocols. J. Cryptogr. Eng. **3**(1), 17–28 (2013)
102. Rührmair, U., van Dijk, M.: PUFs in security protocols: attack models and security evaluations. In: IEEE Symposium on Security and Privacy 2013, pp. 286–300 (2013)
103. Rührmair, U., Sölter, J., Sehnke, F.: On the foundations of physical unclonable functions. Cryptology e-Print Archive (2009)
104. Rührmair, U., Busch, H., Katzenbeisser, S.: Strong PUFs: models, constructions and security proofs. In: Sadeghi, A.-R., Tuyls, P. (eds.) Towards Hardware Intrinsic Security: Foundation and Practice. Springer, Berlin (2010)
105. Rührmair, U., Jaeger, C., Hilgers, C., Algasinger, M., Csaba, G., Stutzmann, M.: Security applications of diodes with unique current-voltage characteristics. In: Financial Cryptography and Data Security (FC). Lecture Notes in Computer Science, vol. 6052, pp. 328–335. Springer, Berlin (2010)
106. Rührmair, U., Sehnke, F., Sölter, J., Dror, J., Devadas, S., Schmidhuber, J.: Modeling attacks on physical unclonable functions. In: ACM Conference on Computer and Communications Security (ACM CCS), pp. 237–249 (2010)
107. Rührmair, U., Jaeger, C., Algasinger, M.: An attack on PUF-based session key exchange and a hardware-based countermeasure: erasable PUFs. In: Financial Cryptography and Data Security 2011, pp. 190–204 (2011)
108. Rührmair, U., Jaeger, C., Bator, M., Stutzmann, M., Lugli, P., Csaba, G.: Applications of high-capacity crossbar memories in cryptography. IEEE Trans. Nanotechnol. **10**(3), 489–498 (2011)
109. Rührmair, U., Devadas, S., Koushanfar, F.: Security based on physical unclonability and disorder. In: Tehranipoor, M., Wang, C. (eds.) Introduction to Hardware Security and Trust, pp. 65–102. Springer, New York (2012)
110. Rührmair, U., Hilgers, C., Urban, S., Weiershäuser, A., Dinter, E., Forster, B., Jirauschek, C.: Optical PUFs reloaded. IACR Cryptology ePrint Archive, Report 2013/215 (2013)
111. Rührmair, U., Sölter, J., Sehnke, F., Xu, X., Mahmoud, A., Stoyanova, V., Dror, G., Schmidhuber, J., Burleson, W., Devadas, S.: PUF modeling attacks on simulated and silicon data. IEEE Trans. Inf. Forensics Secur. **8**(11), 1876–1891 (2013)
112. Rührmair, U., Schlichtmann, U., Burleson, W.: Special session: how secure are PUFs really? On the reach and limits of recent PUF attacks. In: Design, Automation and Test in Europe (DATE) (2014)
113. Rührmair, U., Xu, X., Sölter, J., Mahmoud, A., Majzoobi, M., Koushanfar, F., Burleson, W.: Efficient power and timing side channels for physical unclonable functions. In: CHES 2014, pp. 476–492 (2014)
114. Schrijen, G.J., van der Leest, V.: Comparative analysis of SRAM memories used as PUF primitives. In: Design, Automation and Test in Europe (DATE), pp. 1319–1324 (2012)
115. See http://www.informatik.uni-trier.de/~Ley/db/conf/ches/index.html
116. See http://www.informatik.uni-trier.de/~LEY/db/conf/host/index.html
117. See http://www.gi-de.com/en/trends_and_insights/banknote_circulation/\life_of_a_banknote/life-of-a-banknote.jsp
118. See http://rmaes.ulyssis.be/pufbib.php
119. See http://www.answers.com/topic/certegy-inc-1
120. See http://www.adnas.com/products/signaturedna
121. See http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1315910/

122. See http://www.polestarltd.com/ttg/isspeeches/pisec03/index.html
123. See http://www.date-conference.com/conference/session/4.3
124. See http://www.date-conference.com/conference/session/12.2
125. See http://www.date-conference.com/category/session-types/tutorial
126. See www.nxp.com/documents/other/75017366.pdf
127. See http://www.nxp.com/news/press-releases/2013/02/nxp-strengthens-smartmx2-security-chips-with-puf-anti-cloning-technology.html
128. See http://investor.microsemi.com/releasedetail.cfm?ReleaseID=731250
129. See http://www.microsemi.com/products/fpga-soc/soc-fpga/smartfusion2
130. See http://www.informatik.uni-trier.de/~LEY/db/conf/date/index.html
131. See http://www.informatik.uni-trier.de/~Ley/db/conf/dac/index.html
132. See http://en.wikipedia.org/wiki/Pollen#mediaviewer/File:Lilium_auratum_-_pollen.jpg
133. See http://en.wikipedia.org/wiki/Filter_paper#mediaviewer/File:Filter_paper_840_3x3_copy.jpg
134. See http://de.wikipedia.org/wiki/Compact_Disc#mediaviewer/Datei:REM_CD_GEPRESST.jpg
135. Shamir, A., van Someren, N.: Playing hide and seek with stored keys. In: Financial Cryptography, pp. 118–124. Springer, Berlin (1999)
136. Sharma, A., Subramanian, L., Brewer, E.A.: PaperSpeckle: microscopic fingerprinting of paper. In: ACM Conference on Computer and Communications Security (ACM CCS), pp. 99–110 (2011)
137. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997)
138. Simmmons, G.J.: Identification of data, devices, documents and individuals. In: Annual International Carnahan Conference on Security Technology, pp. 197–218 (1991)
139. Simons, P., van der Sluis, E., van der Leest, V.: Buskeeper PUFs, a promising alternative to D Flip-Flop PUFs. In: HOST 2012, pp. 7–12 (2012)
140. Skorobogatov, S.: Low temperature data remanence in static RAM. Technical Report, UCAM-CL-TR-536, Computer Laboratory, University of Cambridge (2002). Available from http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-536.pdf
141. Smith, J.R., Sutherland, A.V.: Microstructure based indicia. In: Second Workshop on Automatic Identification Advanced Technologies, pp. 79–83 (1999)
142. Sreedhar, A., Kundu, S.: Physically unclonable functions for embeded security based on lithographic variation. In: Design Automation Conference (DAC), pp. 1632–1637 (2011)
143. Stepney, S.: Journeys in non-classical computation. In: Hoare, T., Milner, R. (eds.) Grand Challenges in Computing Research, pp. 29–32. BCS, Swindon (2004)
144. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Design Automation Conference (DAC), pp. 9–14 (2007)
145. Suh, G.E., Clarke, D.E., Gassend, B., van Dijk, M., Devadas, S.: AEGIS: architecture for tamper-evident and tamper-resistant processing. In: Proceedings of the 17th International Conference on Supercomputing (ICS), pp. 160–171 (2003)
146. Suzuki, D., Shimizu, K.: The glitch PUF: a new delay-PUF architecture exploiting glitch shapes. In: Proceedings of CHES 2010, pp. 366–382 (2010)
147. Troncoso, C., De Cock, D., Preneel, B.: Improving secure long-term archival of digitally signed documents. In: Kim, Y., Yurcik, B. (eds.) Proceedings of the 4th International Workshop on Storage, Security and Survivability (StorageSS 2008), pp. 27–36 (2008)
148. Tuyls, P., Skoric, B., Stallinga, S., Akkermans, A.H.M., Ophey, W.: An information theoretic model for physical uncloneable functions. In: IEEE International Symposium on Information Theory, p. 141 (2004)
149. Tuyls, P., Skoric, B., Stallinga, S., Akkermans, A.H.M., Ophey, W.: Information-theoretic security analysis of physical uncloneable functions. In: Financial Cryptography, pp. 141–155 (2005)
150. Tuyls, P., Schrijen, G.J., Skoric, B., van Geloven, J., Verhaegh, N., Wolters, R.: Read-proof hardware from protective coatings. In: Proceedings of CHES 2006, pp. 369–383 (2006)

151. Vaidya, A.W.: Keeping card data secure at low cost. In: European Convention on Security and Detection, pp. 212–215 (1995)
152. van der Leest, V., Preneel, B., van der Sluis, E.: Soft decision error correction for compact memory-based PUFs using a single enrollment. In: Proceedings of CHES 2012, pp. 268–282 (2012)
153. van der Leest, V., Tuyls, P.: Anti-counterfeiting with hardware intrinsic security. In: Design, Automation and Test in Europe (DATE), pp. 1137–1142 (2013)
154. van Dijk, M.: System and method of reliable forward secret key sharing with physical random functions. US Patent No. 7,653,197 (2004)
155. van Renesse, R.L.: 3DAS: a 3-dimensional-structure authentication system. In: European Convention on Security and Detection, pp. 45–49. Institution of Electrical Engineers, London (1995)
156. van Renesse, R.L.: Optical Document Security, 3rd edn. Artech House (2005) [ISBN-10: 1580532586]
157. Vijayakumar, A., Kundu, S.: A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics. In: Design, Automation and Test in Europe (DATE 2015), pp. 653–658 (2015)
158. Vijaywargi, D., Lewis, D., Kirovski, D.: Optical DNA. In: Financial Cryptography 2009, pp. 222–229 (2009)
159. Wang, X., Tehranipoor, M.: Novel physical unclonable function with process and environmental variations. In: Design, Automation and Test in Europe (DATE), pp. 1065–1070 (2010)
160. Wikipedia's article on cryptography (2012). Downloaded from http://en.wikipedia.org/wiki/Cryptography, August 2012
161. Wikipedia's article on data remanence (2012). Downloaded from http://en.wikipedia.org/wiki/Data_remanence, August 2012
162. Wikipedia's article on a one-electron universe (2014). Downloaded from http://en.wikipedia.org/wiki/One-electron_universe, April 2014
163. Yamamoto, D., Sakiyama, K., Iwamoto, M., Ohta, K., Ochiai, T., Takenaka, M., Itoh, K.: Uniqueness enhancement of PUF responses based on the locations of random outputting RS latches. In: CHES 2011, pp. 390–406 (2011)
164. Yao, Y., Kim, M., Li, J., Markov, I.L., Koushanfar, F.: ClockPUF: physical unclonable functions based on clock networks. In: DATE 2013, pp. 422–427 (2013)
165. Yin, C.-E.D., Qu, G.: Improving PUF security with regression-based distiller. In: Design Automation Conference (DAC), paper no. 184, 6 p. (2013)
166. Yin, C.-E.D., Qu, G., Zhou, Q.: Design and implementation of a group-based RO PUF. In: Design, Automation and Test in Europe (DATE), pp. 416–421 (2013)
167. Yu, M.-D., M'Raihi, D., Sowell, R., Devadas, S.: Lightweight and secure PUF key storage using limits of machine learning. In: Proceedings of CHES 2011, pp. 358–373 (2011)
168. Zheng, Y., Hashemian, M., Bhunia, S.: RESP: a robust physical unclonable function retrofitted into embedded SRAM array. In: Design Automation Conference (DAC), paper no. 60, 9 p. (2013)

Design and Implementation of High-Quality Physical Unclonable Functions for Hardware-Oriented Cryptography

Siarhei S. Zalivaka, Le Zhang, Vladimir P. Klybik, Alexander A. Ivaniuk, and Chip-Hong Chang

Abstract Physical Unclonable Functions have emerged as effective primitives for varieties of security applications. With the advent of mobile computing, designing and implementing high-quality PUFs for resource-constrained platforms become a great challenge. This chapter presents an extensive review of the techniques proposed in the recent years for the design and implementation of high-quality and/or alternative PUF instances with marginal overhead. With a preamble of the motivations, fundamentals, quality metrics and application scenarios of PUF, some existing approaches to improving the quality of PUFs are unfolded. Subsequently, some representative PUF designs for RFIDs and fingerprint extractions are illustrated. In addition, applications for true random number generation based on PUF instances are delineated. Finally, two emerging types of PUF implementations that can be used for more advanced protocols are presented. The practices summarized in this chapter aim to help the engineers and researchers in the hardware security community to design and implement PUFs that suit their applications and constraints.

1 Introduction

The proliferation of connected mobile devices has boosted the growth of consumer web and services of all types. Driven by economies of scale, a large scale distributed computing paradigm has emerged, in which a cloud of abstracted virtualized, dynamically-scalable, distributed computing power, storage and services are delivered online and on demand to consumers over the internet. As electronic devices become ubiquitous and interconnected, people are increasingly relying on

S.S. Zalivaka • L. Zhang • C.-H. Chang (✉)
School of Electrical and Electronic Engineering, Nanyang Technological University, 639798 Singapore
e-mail: zali0001@e.ntu.edu.sg; LZHANG15@e.ntu.edu.sg; ECHChang@ntu.edu.sg

V.P. Klybik • A.A. Ivaniuk
Department of Computer Science, Belarusian State University of Informatics and Radioelectronics, P. Browki Str., 6 Minsk 220013, Belarus
e-mail: klybik@bsuir.by; ivaniuk@bsuir.by

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_2

integrated networked devices to perform security sensitive tasks as well as handling large amount of confidential digital information. The association of internet and infocomm technology has created tremendous economic values that even a limited form of exploitation can yield enormous returns or losses. Responding to this nascent shift in the volume of valuable information traffic is a host of up and coming cyber-terror and hackings. The cryptography community has recently expressed concerns about whether the software-oriented security alone is adequate to combat the imminent threats of the cyberspace, the largest ever interdependent network of information system.

Physical cryptography has emerged as an effective approach to solving issues such as the secure key storage, a problem that has troubled the designers of crypto-system for ages with the conventional cryptography. Instead of relying on certain specifically safeguarded components that may be easily invaded via physical means to preserve the secret keys, physical cryptography exploits the inherent non-reproducible characteristics of electronic devices to guarantee the physical unbreakability of crypto-system. One of the promising representatives of physical cryptographic primitives is the Physical Unclonable Function (PUF). The feasibility of PUFs stems from the fact that most devices fabricated using sub-micron or nano-scale technologies are highly susceptible to the impacts of process variations. That is, the physical parameters of devices after the manufacturing process are often stochastically distributed. The physical parameters may refer to different kinds of physical effects depending on the media used for implementing the PUF, e.g., speckle pattern of optically sensitive material [58], frequencies of oscillating structures [24], transmission-speed difference of racing signals in delay paths [43], property disparities of coating layers [61], randomness of initial state of memory or memory-like circuitry [29]. While these parametric variations appear to be a fundamental evil for useful circuit design in general, they turn out to be an excellent entropy source for cryptographic primitives to generate device-specific secret bits. Normally a randomly selected stimulus is applied on the PUF devices to produce a random output signal. The stimulus is called the "challenge" while the output signal is called the "response". To make a PUF compatible with the contemporary integrated system, challenges and responses are normally digitized and represented as binary bits.

The desiderata of a good PUF are: firstly, its responses should be random so that they cannot be predicted by the adversaries; secondly, its responses to the same challenges must be reproducible over time, and the reproducibility of responses should be maintained under all possible working conditions; lastly, the responses generated from a device using different challenges or from different devices using the same challenge must be unique and distinctive from each other.

Though the recent years have witnessed a number of new PUF implementations, some emerging security problems, implementation and technical challenges are awaiting for better solutions. The problems may arise from the stringent requirements of PUF quality under varying operating conditions and/or the security level of the cryptographic protocol. Some of these challenges have been partially addressed by the new development of PUF but they remain the stumbling blocks since its inception, while others become teething problems with the advance of modern communication technologies such as wireless network, internet of things, and mobile computing where the area and power budget is thinning aggressively. To this end, the interest of PUF research has shifted from merely "looking for the feasibility of designing a PUF" at the beginning to "looking for a better medium/method/structure/protocol of designing and using a PUF". More specifically, the new focus can be viewed from two aspects:

1. **Enhancing the response quality**. Techniques at architecture-level such as the application of Error Correction Code (ECC), fuzzy response analysis, post-processing using hash function, etc. have been proposed and widely used by the community to improve the qualities of PUFs, but device- and circuit-level engineering techniques that achieve the same goal with lower area/power cost are rarely exploited. Second-order characteristics of CMOS devices (e.g., aging effects, thermal characteristics, etc.) as well as the emerging technologies based on alternative nano-materials may help to realize this goal.
2. **Implementing and designing PUF for new applications.** Novel PUF instances such as public PUF, reconfigurable PUF and new PUF architectures were proposed to be incorporated in more advanced protocols, where PUFs are endowed with new features to be used as more powerful primitives. Compared to conventional PUFs, these types of PUFs may have more sophisticated structures, thus more dedicated design and analysis methods are required to prove the efficiency and effectiveness.

The remainder of the chapter is organized as follows. Section 2 presents the PUF fundamentals and an overview of the basic concepts of PUF hardware identification, authentication and main areas of application. Section 3 introduces several techniques to improve the reliability of PUFs. These include architecture-level methods such as fuzzy response analysis and application of ECC, and circuit-level design techniques such as temperature-aware stability compensation in ring-oscillator PUF and reliability enhancement in memory-based PUFs. Hybrid PUF architectures will also be discussed in this section. Section 4 reviews some representatives of classical PUF implementations. General scheme and commercial implementation of RFID crypto-processor are introduced. FPGA IP-protection approaches and their experimental results are presented. Dual-mode PUF circuits for true random number and unique ID generation are also discussed. Section 5 presents new derivatives of PUFs, where reconfigurable and public PUF are discussed. The chapter is concluded in Sect. 6.

2 Physical Unclonable Function (PUF) Fundamentals

Physical Unclonable Function (PUF) is also called Physical One-Way Function (POWF) and Physical Random Functions (PRF) [24, 25, 58, 59]. Although the two latter terms were introduced earlier, the term Physical Unclonable Function (PUF), coined by Tuyls et al. in [83], is more commonly used today. However, the notion of "Physical One-Way Function" introduced by Pappu in [58, 59] definitely owns the merit of being the first formal attempt to define the deterministic physical interaction between a probe and system in some unknown state as a primitive for physical cryptography.

2.1 Definitions

Two variants of PUF definition are widely considered:

Definition 1. PUFs consist of inherently unclonable physical systems. They inherit their unclonability from the fact that they consist of many random components that are present in the manufacturing process which cannot be controlled. When a stimulus is applied to the system, it reacts with a response. Such a pair of stimulus (challenge) C and response R is called a challenge-response pair (CRP). In this respect, a PUF can be considered as a function that maps challenges to responses.

According to this definition, PUFs have been introduced as challenge response entities that are undetachably embedded in a physical system [40], possibly a silicon integrated circuit. The physical system is, explicitly or implicitly, produced in such a way that it contains uncontrollable random elements. An applied challenge reacts with these elements in a complex and unpredictable way to produce a certain random response. The presence of random elements and the impossibility of controlling them at manufacturing time make each PUF unique and physically unclonable. A more general definition of PUFs as Super High Information Consent systems (SHICs) has also been made by Ruhrmair [71].

Definition 2. Physical Unclonable Functions are complex disordered physical systems with extraordinarily high amount of structural information that satisfy the following conditions:

– The information content of the system can be extracted reliably and repeatedly through measurement with different challenge (interrogation) C_i to obtain the resulting response (answer) R_i.
– The number of possible challenges C is so large that the values of all corresponding responses R cannot be determined for all possible challenges C within limited time.
– Due to the very high information content of the system, it must also be impossible to model, computationally learn, simulate or otherwise numerically predict the challenge-response pair (C_j, R_j) based on the known pair (C_i, R_i).
– It must be prohibitively difficult to physically reproduce or clone a PUF although the physical system itself is easy to make.

The fundamental idea behind many implementations for the silicon based PUFs is to create an integrated circuit whose output will be determined by the timing deviation

Table 1 Definitions of used notations

Notation	Definition
N	The number of devices.
K	The number of different IDs generated per device.
T	The number of tests performed per ID.
L	The length of an ID.
n	The index of a device. $1 \le n \le N$. The n-th device is denoted by n for simplicity.
k	The index of an ID. $1 \le k \le K$. The k-th ID is denoted by k for simplicity.
t	The index of a test. $1 \le t \le T$. The t-th test is denoted by t for simplicity.
l	The bit position of an ID. $1 \le l \le L$. The l-th bit is denoted by l for simplicity.
$ID_{n,k}$	The correct ID k expected to be generated in device n.
$ID_{n,k,l}$	The empirically generated ID k by device n in test t.
$b_{n,k,l}$	The correct response bit l of ID k expected to be generated by device n. $b_{n,k,l} \in \{0, 1\}$
$b_{n,k,t,l}$	The empirically generated response bit l of ID k by device n in test t. $b_{n,k,t,l} \in \{0, 1\}$

resulting from the underlying silicon device variation. Due to the variation of the imperfect scaled silicon manufacturing process, the timing of certain paths through the devices will vary slightly from chip to chip [1].

2.2 Metrics of Evaluation: Randomness, Stability and Uniqueness

For practical implementations, it is necessary to quantify and measure the figures of merit of PUF. The most important of all are the randomness, stability and uniqueness. These metrics are estimated or measured from the physical manifestation of the desired properties of PUF. They are useful for the purpose of benchmarking and assessing the acceptability of an implemented PUF instance for a given application or to attain certain security strength. For the ease of reference and exposition, the notations used in the definitions of these metrics are listed in Table 1.

2.2.1 Correct IDs

Suppose a PUF in device n outputs K different IDs from K different challenge sets. Let $ID_{n,k}$ be the correct ID k generated in device n. $ID_{n,k}$ is empirically determined through T tests where the same ID k is to be generated T times.

Let $ID_{n,k,t}$ be the ID k generated in device n in test t, and $b_{n,k,t,l} \in \{0, 1\}$ be the l-th bit value of $ID_{n,k,t}$. Then, $ID_{n,k,t}$ can be expressed as follows:

$$ID_{n,k,t} = b_{n,k,t,1} \parallel b_{n,k,t,2} \parallel \cdots \parallel b_{n,k,t,L} \tag{1}$$

where $\parallel$ is an associative operator denoting the concatenation of two operands.

Let $p_{n,k,l}$ be the relative frequency that the l-th bit of $ID_{n,k,t}$ is 1 in a test. $p_{n,k,l}$ is defined as the average number of 1's in $ID_{n,k,t}$ generated in T tests, i.e.,

$$p_{n,k,l} = \frac{1}{T}\sum_{t=1}^{T} p_{n,k,t}, \tag{2}$$

Let $b_{n,k,l}$ be the l-th bit of the correct ID, $ID_{n,k}$. Then $b_{n,k,l}$ is defined as follows:

$$b_{n,k,l} = \lfloor(p_{n,k,l} + 0.5)\rfloor = \begin{cases} 1 & \text{if } p_{n,k,l} \geq 0.5 \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

With the definition of $b_{n,k,l}$, the correct ID, $ID_{n,k}$ is defined as follows:

$$ID_{n,k} = b_{n,k,1} \parallel b_{n,k,2} \parallel \cdots \parallel b_{n,k,L} \tag{4}$$

2.2.2 Randomness

A PUF is expected to output 0 and 1 ideally with the same probability. Randomness indicates the balance of 0s and 1s in the responses of the PUF.

Let p_n be the relative frequency of 1 appearing in all the response bits generated in device n. The p_n is given by:

$$p_n = \frac{1}{K \cdot T \cdot L}\sum_{k=1}^{K}\sum_{t=1}^{T}\sum_{l=1}^{L} b_{n,k,t,l} \tag{5}$$

The device randomness H_n is defined by the min-entropy, which is suitable for evaluating the randomness of a bit sequence.

$$H_n = -log_2 max(p_n, 1 - p_n) \tag{6}$$

The undefined $log_2(0)$ is set to 0. H_n takes the highest value 1 when $p_n = 0.5$, and the lowest value 0 when $p_n = 0$ or $p_n = 1$.

2.2.3 Stability

When the same ID is generated T times in the same device, the l-th bits of the output IDs are expected to be identical. The stability is an indicator of how stable is an output bit in the PUF response to the same challenge, or in other words, how strongly is $p_{n,k,l}$ biased towards 0 or 1.

Let $S_{n,k,l}$ be the stability of the l-th bit of $ID_{n,k}$. Then $S_{n,k,l}$ is defined by using min-entropy as follows:

$$S_{n,k,l} = 1 + log_2 max(p_{n,k,l}, 1 - p_{n,k,l}) \tag{7}$$

with the undefined $log_2(0)$ set to 0. Then, the device stability S_n is defined by taking the mean of $S_{n,k,l}$ as follows:

$$S_n = \frac{1}{K \cdot L} \sum_{k=1}^{K} \sum_{l=1}^{L} S_{n,k,l} = 1 + \frac{1}{K \cdot L} \sum_{k=1}^{K} \sum_{l=1}^{L} log_2 max(p_{n,k,l}, 1 - p_{n,k,l}) \tag{8}$$

S_n takes the highest value 1 when $p_{n,k,l} = 0$ or $p_{n,k,l} = 1$, and the lowest value 0 when $p_{n,k,l} = 0.5$.

2.2.4 Uniqueness

When the same challenge sets are given to different PUFs, the output IDs are expected to be different. Uniqueness indicates how different are the generated IDs among the devices. Since uniqueness is an inter-device characterization, all the possible device combinations should be considered. As proven in [31], the sum of the Hamming distance of the possible ID-combinations among all devices does not exceed $L \cdot (N/2)^2$. Based on this study, the device uniqueness U_n is defined as follows:

$$U_n = \frac{4}{K \cdot L \cdot N} \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{j=1, j \neq n}^{N} (b_{n,k,l} \oplus b_{j,k,l}) \tag{9}$$

Then, the sample mean $\bar{U}$ is defined as

$$\bar{U} = \frac{1}{N} \sum_{n=1}^{N} U_n = \frac{4}{K \cdot L \cdot N^2} \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (b_{i,k,l} \oplus b_{j,k,l}) \tag{10}$$

$\bar{U}$ takes the highest value 1 when the sum of the Hamming distance of the possible ID-combinations reaches its asymptotical upper bound, and the lowest value 0 when the IDs in device n are completely identical.

2.3 PUF for Identification and Authentication

2.3.1 Hardware Identification

Identification is the act of finding out who someone is or what something is. For the digital devices, this procedure is used in different aspects, and in most cases

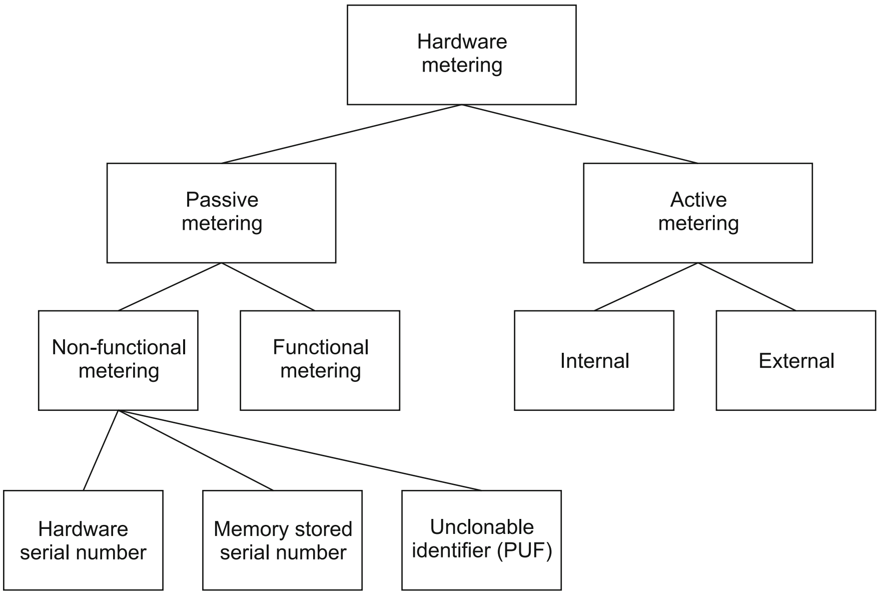


Fig. 1 Hardware metering classification

to address the question: "what is the identity of this device?" as an attempt to find out indirectly who owns the device. This device tagging problem has been encountered and solved in the manufacturing, sale, maintenance and operation of various hardwares. Figure 1 shows a classification of hardware identification methods.

Previously, the identification problem was solved by the hardware pre-programmed digital ID, for example, RFID product identification [44] and battery packs authenticity check chip [22]. The implementation required a separate phase for recording the IDs and the adoption of special measures to ensure their uniqueness.

With a properly designed PUF, the output response is unique for each physical copy of the work unit [84]. This property allows the use of PUF to obtain a unique device identifier during its production phase without the need to introduce customization for each instance. Using PUF to identify the device as built can reduce costs in mass hardware production [46].

2.3.2 Device Authentication

Besides identification, digital device authentication is also required in many applications [10, 60]. Unlike identification, for which the purpose is to find the desired device, authentication is to confirm that the device is what it claimed to be, i.e., the confirmation of its authenticity.

Device authentication demands more than just the interaction between the two devices in communication. In some case, the device is also required to perform certain operations to validate its integrity and origin. The digital device that is endowed with this capability is said to be self authenticable. One of the most common attacks on digital devices, which are implemented on the FPGA, is the cloning of bitstream. Due to the fact that the original circuit implementation description (netlist, VHDL-codes) and bitstreams for FPGA configuration are presented in an open format, the attacker can easily replicate them without much effort. If left unprotected, the attacker can also reverse engineer the plain FPGA bitstream to obtain the netlist and figure out the functionality to tamper with the digital circuit. More often than not, the bitstream was effortlessly cloned (e.g. for implementation on cheaper FPGA) as a "black box", without analyzing and changing its operation.

One of the options to protect digital bitstreams of FPGA devices against unauthorized usage is to use conventional hardware cryptographic primitives, in particular for the authentication of pre-programmed digital ID. Device authentication based solely on external cryptographic components may not be adequate in view of the potential technical failure or intentionally released duplicate devices that are indistinguishable from the original. Failure or vulnerability in device authentication can be critical when the authenticated device is used in:

- control access to important objects or infrastructure,
- IP protection,
- interaction of mission-critical equipments.

Such duplication is impossible for PUF-based digital device authentication because the device authenticity is determined by the innate variations of the production process that cannot be deterministically controlled.

2.3.3 Area of Applications

The main PUF applications lie in the tasks of enhancing the security of digital devices themselves and processes associated with their use. The following are some applications of PUF in the identification and authentication of digital devices. Each of them has subtly different requirements on the properties of PUF and certain types of PUF are preferred in some areas.

- True random number sequence generation
- Secret key generation
- Preventing IC piracy and digital rights management
- Hardware Trojan and recycled IC detection
- Product authentication in supply chain applications
- Unique serial numbering
- Shooter authentication for weapon control
- ID and authentication for self-destructing electronics

Truly random sequences is used to create encryption keys for digital signatures, as session keys for message encryption in communication, and as password to access the protected systems. If the sequence is not truly random, but pseudorandom, it will adversely weaken the reliability of the protection scheme and may render the entire system security to be broken [43].

Identifying malicious hardware implants (Trojans) is a very time-consuming and non-trivial task for hardware developers and for security professionals. Using PUF enables the features associated with the physical characteristics of a genuine device to be extracted in the form of PUF responses and used as a set of reference signatures to winnow the Trojan-infected chips from the good chips. For example, the addition of the hardware implant changes the distribution of power profile, resulting in a distinguishable deviation in the response profile of the on-chip PUF [99].

The profile of PUF responses changes with the passage of time due to device aging. The use of special algorithms to analyze the response characteristics of PUF help to dispose used hardware that has degraded performances or hardware that requires replacement due to natural aging. Detection of such equipment is an important task in critical applications in aviation, military industry, medical equipment, etc., where degraded or recycled parts or equipment are endangered threat to human life or cause of man-made disasters.

Authentication of products in supply networks is important to protect the original manufacturers of the losses due to the contaminated supply chain. Counterfeit products undermine company's brand and impact customer loyalty. Consumers need to acquire authentication of their devices for protection against counterfeiting when applying to the service center for warranty. Production of counterfeit goods has become a big problem in physical products as well as in the content. In both cases, there is a dire need to protect the intellectual property (IP) rights of the authors and developers [5, 67]. For content protection, Digital Rights Management (DRM) is used [39]. The basis of this system is a content playback device authentication and a cryptographic binding of the content to specific devices. The PUF implemented in the playback device forms the basis for the authentication of such devices in the DRM system.

An apt application of PUF is in hand weapon authentication. Weapon embedded authentication device is used for additional security and liability associated with the use of a weapon to identify the weapon to its authorized users. In case of loss or theft of weapons, the attacker cannot use it against the owner or other people [2].

New direction for using PUF was explored in the creation of self-destruction electronics. This type of electronics is of great demand in the military and defense sectors to protect secrets and techniques contained in devices left in the battlefield and captured by the enemy. A special signal can be sent to activate the device self-destruction mechanisms without the possibility of recovery. Authentication is mandatory to avoid false alarms as the enemy could also send a similar signal to permanently destruct or disable the device [20]. In commercial applications, PUFs can also be used to authenticate and disable devices to limit the lifespan of the devices or upon expiry of the license agreement.

3 Quality Enhancement of PUF

One of the main concerns of using PUF for authentication is its raw response instability. There are a lot of techniques to make PUF responses more reliable in practice. On the other hand, the response instability may be beneficial for certain applications like true random number generation.

3.1 PUF Response Analysis and Fuzzy Identification

Usually, there is no stable challenge-response (CRP) pair in real PUF implementation. Bit error rate (BER) is used to describe PUF response errors. It is a ratio of the number of error bits to the length of the response. Without any post-processing, PUF responses with BER of about 10 % or more is common [78]. The probability of response can be computed by

$$P(r = 1, n) = \frac{\sum\limits_{i=1}^{n} r_i}{n}, \tag{11}$$

where n is number of experiments and r_i is the value of response after the i-th experiment.

Thus, a PUF can also be represented as a fuzzy set $CRP_F(n) = \{(c_1, \{p(r_1 = 0, n)/r_1, p(r_1 = 1, n)/r_1\}), (c_2, \{p(r_2 = 0, n)/r_2, p(r_2 = 1, n)/r_2\}, \ldots, (c_k, \{p(r_k = 0, n)/r_k, p(r_k = 1, n)/r_k\}\}$, where k is the number of challenges, and single-bit response is assumed. If the response is multi-bit, the probabilities of every possible response values will be calculated and the challenge fuzzy-set will have greater cardinality. The fuzzy set $CPR_F(n)$ can be used as the digital device identifier. However, as storing and processing this data is not as fast and simple, the following steps are needed to compute the ID.

- **Defuzzification**. This procedure transforms the fuzzy set CRP_F into the ordinary CRP. The values of the transformed CRP may be different from the original. For example, all CRP_F values may be divided into five groups: zero (Z) for $p(r, n) = 0$, almost zero (AZ) for $0 < p(r, n) < 0.3$, unstable (U) for $0.3 \leq p(r, n) \leq 0.7$, almost one ($AO$) for $0.7 < p(r, n) < 1$ and one (O) for $p(r, n) = 1.0$. Thus, CRP_F will be transformed into $CPR = \{(c_1, r_1^s), (c_2, r_2^s), \ldots, (c_k, r_k^s)\}$, where $r_i^s \in R = \{Z, AZ, U, AO, O\}$. If almost all the response values are stable (i.e., more than 60 % of the values belong to Z or O), it is possible to use a simpler set $R = \{Z, U, O\}$, where $U = AZ$, U or AO. Finally, the ID can be computed by the majority rule in Eq. (12).

$$r_i^s = \begin{cases} 0, p(r_i, n) < 0.5 \\ 1, p(r_i, n) \geq 0.5 \end{cases} \tag{12}$$

– **Signature Analysis**. The complexity of comparing, storing and processing k-bit identifiers, where $k = 2^m$ and m is the number of bits of the challenge, grows exponentially with the challenge length m. To transform the set *CRP* into a digital device ID, signature analysis is used, where the signature is obtained by means of some lossy data compression algorithm. The classic method of signature analysis is based on the polynomial division over the Galois field of two elements [91].

There are three possible ways of fuzzy identification:

– **Comparing Probabilities**. The digital identifiers represented by the sets CRP_F is compared directly. If the absolute difference of the probabilities of every challenge-response pair is less than $\frac{3}{n}$, it is sufficient to identify the digital device. This approach has a great disadvantage that all operations with the set CRP_F involve floating-point numbers, which are slow and inefficient. This approach may be implemented if the responses are unstable and $n \geq 1{,}000$.
– **Comparing Multiple Values**. If the responses are more stable, the probabilities can be transformed into the values from the set R. To process these values, only 2 or 3 bits are needed. Nevertheless, if the cardinality of the set CRP_F is relatively big ($\geq$1,000), it will be compressed by the signature analysis algorithm. This approach has moderate rate and reliability.
– **Comparing Binary Values**. Finally, it is possible to process digital identifiers which are computed by majority rule or signature analysis. The *CRP* sets can be compared by their Hamming distances. Small distances (0–3 % of k) imply true identification and greater distances imply false identification. The signatures have to be compared directly because its size is much smaller than the cardinality of the set *CRP*.

Thus, the fuzzy identification algorithm contains four main steps: collection of challenge response pairs (n times), computation of the set CRP_F, different ways of creating the *CRP* set and signature analysis to obtain the set *CRP* (if needed).

3.2 Application of ECC

To improve the quality of the CRP_F set, different Error Correction Codes (ECCs) are used. An ECC is used to correct the errors after data transmission. In this case, the PUF is treated like a data channel, where the "ideal" (i.e., the most probable) *CRP* set is a message that is transmitted through the noisy PUF channel. After the correction, the corrected set CRP_C will be as close as possible to the "ideal" set *CRP*.

The ECC approach is evaluated by three criteria:

– **Redundancy**. This is a ratio of the total number of bits (response + helper data) to the length of the response. Codes with low redundancy is easier to implement and faster to process but its error correction capability is limited. If the response reliability is more important, the redundancy of the ECC may be increased.
– **Complexity**. The computational time and hardware resource requirements of different ECC algorithms are different. Some approaches are harder to implement in hardware. Choosing an appropriate method is a trade-off between the quality and implementation cost.
– **Efficiency**. One of the most important features of ECC is to improve the BER. The efficiency can be represented as a ratio of the BERs before and after the correction. Reduction of BER is almost always necessary to implement a reliable design unless the error rate of the raw response is tolerable by the application.

Some of the most widely used ECC techniques include Hamming code [28], repetition code [45, 56], **H**ocquenghem and Raj Chandra **B**ose and Dwijendra Kumar Ray-**C**haudhuri code (BCH code) [55] and the newest one—soft decision code [93].

Repetition Code is the simplest ECC. The main idea is to repeat the PUF responses n times and the final PUF response is obtained by majority voting. It should be noted that this approach has high redundancy, but reasonably good reliability. The BER after correction is very low. This approach has been used in some PUF implementations [9]. Their investigation shows that PUF responses with high initial BER ([0.2,0.5] and higher) are not very well corrected by different repetition codes (Rep(11), Rep(21) and Rep(31)). However, if the BER is not that high, its effectiveness increases linearly with increasing n. The main advantage of this ECC technique is its implementation simplicity and high corrected PUF response reliability with BER less than 0.2. Some kind of repetition code is also used for fuzzy identification (see Sect. 3.1 below).

Hamming Code is a more complex ECC. To detect up to two-bit errors and correct one-bit errors in some binary word, parity bits (or check bits) are embedded. The most widely used Hamming code is Hamming(7, 4), i.e., 4-bit word with 3 parity bits. Different versions of Hamming code are shown in Table 2. The redundancy decreases rapidly with increasing word length. Thus, the unstable bits in PUF response are divided into an appropriate number of groups (binary words) and corrected. Hamming code is not widely used by PUF developers, but there are some investigations of its use [24]. Despite its low capability to correct many errors, it has low redundancy and can be easily implemented.

BCH Code is a cyclic ECC designed based on the primitive polynomial. It can be described by a triple (n, l, d), where n is the length of the codeword, l is the

Table 2 Different versions of Hamming code

Word length	1	4	11	26	57	120	247
Number of parity bits	2	3	4	5	6	7	8

Table 3 Comparison of ECCs

Code	Redundancy (rank)	Complexity (rank)	Efficiency (rank)	Aggregate rank
Repetition	High (3)	Low (1)	High (1)	5
Hamming	Low (1)	Middle (2)	Low (3)	6
BCH	Middle (2)	High (3)	Middle (2)	7

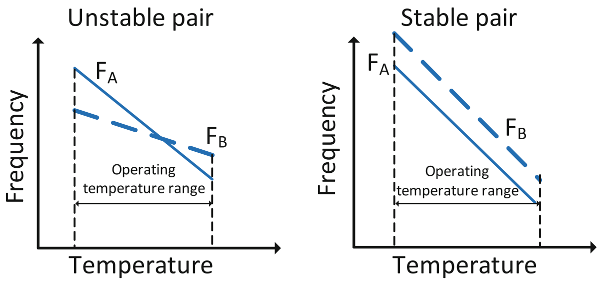


Fig. 2 Frequency versus temperature characteristics of (*left*) unstable and (*right*) stable RO pairs (subscripts A and B refer to the two oscillators of the pair)

length of the original word and d is the minimum Hamming distance between all pairs of words. Thus, $\frac{d-1}{2}$ errors can be corrected by this code. The error correction capability of BCH code is better than Hamming code and the ratio of redundancy to efficiency is better than the repetition code. It has been widely used to correct noisy PUF responses. For example, BCH (127, 64, 21) code was used to correct 64-bit secret keys generated by the ring oscillator PUF (RO-PUF) [79].

The ECCs described above are compared in Table 3. According to Table 3, repetition code has the best overall performance. The choice of a suitable code, however, depends on the priorities of the PUF performances. For example, if low hardware cost is more desirable and fewer errors are allowed, then repetition code with high redundancy may not be the best choice. On the other hand, low error rate with faster implementation and higher hardware cost is a more preferred solution for PUF of high reliability requirements.

3.3 Temperature-Aware PUF

One of the main reasons for PUF response instability is temperature variations. For instance, the response of RO-PUF is obtained by comparing the frequency of a pair of oscillators, but some pairs have almost identical frequency. Therefore, the simplest approach is to choose two oscillators out of a set of $n \geq 4$ oscillators. Due to the temperature variations, the response value of the chosen pair may change. Thus, the RO pairs may be divided into two groups: stable and unstable (see Fig. 2).

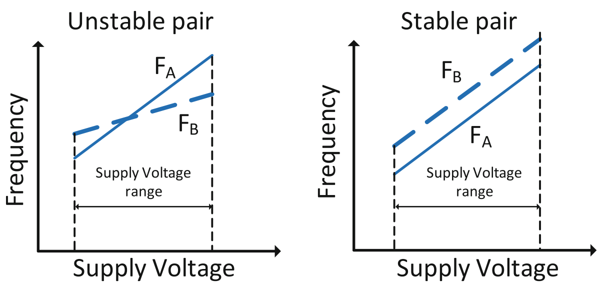


Fig. 3 Frequency versus supply voltage relation of (*left*) unstable and (*right*) stable RO pairs (subscripts A and B refer to the two oscillators of the pair)

Due to the response instability, the RO pairs are categorized into three types [63]:

- **Good Pair**. In this type, RO_A has much higher frequency than RO_B (or vice versa) over the entire operating temperature range. This pair will produce a reliable bit one (or zero) and will be used without any transformations.
- **Bad Pair**. This type will produce unstable bit over the whole range of operating temperatures. It will be disconnected from the power source and will not be used.
- **Contributing Pair**. This type of RO pair will produce unstable bit over some temperature range. The unsafe temperature ranges $[t_{min}, t_{max}]$ for each such pair are stored in a non-volatile memory (NVM).

This approach reduces the redundancy and stablizes the RO responses. To implement this approach, small NVM and temperature sensors are required. The information stored in the memory is useless for the attacker even if he has access to it.

The unstable responses of temperature dependent RO pairs can also be corrected by changing the supply voltage [87]. The correction circuit is composed of three main parts: temperature sensor, reconfigurable ROM and RO-PUF block. The **Temperature sensor** is required for adjusting the supply voltage. Every temperature range from $-15, -5, 5, \ldots, 75\,°\text{C}$ has a corresponding voltage. A **Reconfigurable ROM** is used to store the information during the testing phase. At every temperature point, the required supply voltage for each RO pair to produce a stable response bit is selected. The principle behind tuning the supply voltage to rectify the unstable RO pairs in a chosen temperature range can be explained by the linear relation between the frequency and supply voltage in Fig. 3. The **RO-PUF block** contains N RO chains, each is composed of an odd number of inverters with negative feedback. The authors investigated inverter chains of 5, 11 and 21 inverters and concluded that shorter chain leads to better stability.

A more effective way to build the temperature-aware RO-PUF based on supply voltage modulation is proposed in [52]. Its circuit is depicted in Fig. 4. According to Fig. 4, the PUF challenge determines which pair of RO will be chosen. At the same time, the supply voltage configuration corresponding to the PUF challenge is extracted from the NVM. The configuration is applied to the circuit by PMOS

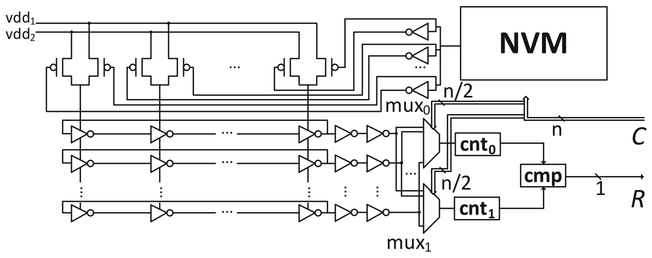


Fig. 4 Schematic of Temperature-aware PUF [52]

switches. The supply voltage for reliable operation in the entire temperature range for every RO pair is stored in the NVM during the testing phase. L voltage levels are stored and the voltage level for each of the C RO columns may be different. Thus, there are L^C different supply voltage combinations. The memory capacity Cap for R RO pairs is given by:

$$Cap = \lceil log_2 L \rceil \cdot C \cdot \frac{R \cdot (R-1)}{2} \tag{13}$$

Leakage of the content of the NVM does not divulge any secret to the attacker. This is because the NVM does not contain the RO frequencies. The supply voltages are pre-computed during the post-manufacturing testing phase. This approach does not require any temperature sensors and requires only halve the memory of [63].

In summary, selecting the RO pair based on its safe temperature range or by changing its supply voltage level is very useful for PUF response stabilization. Very reliable unique ID generator can be constructed by RO-PUF with such kind of pre-processing at low hardware cost.

3.4 Reliability Enhancement of Memory-Based PUFs

Memory-based PUFs (MemPUFs) such as SRAM-based PUF, butterfly PUF, sense-amplifier PUF, flip-flop PUF, buskeeper PUF, or the most recent non-volatile memory based PUFs (e.g., Phase Change Memory (PCM) based PUF and Spin-Transfer Torque (STT) Magnetic RAM (MRAM) based PUF) refer to a group of PUFs that exploit the mismatches of individual memory elements to generate the response bits. Similar to other types of PUFs, MemPUF is also susceptible to the noise effects such that the response bit generated from each cell cannot be reproduced stably over time. For example, if the mismatch of driving ability between the two cross-coupled inverters in an SRAM-based PUF is too small, the response bit generation process can be easily distorted by noise which makes the output bit unreliable. The BERs of most PUFs are as high as 10 %, and can be further

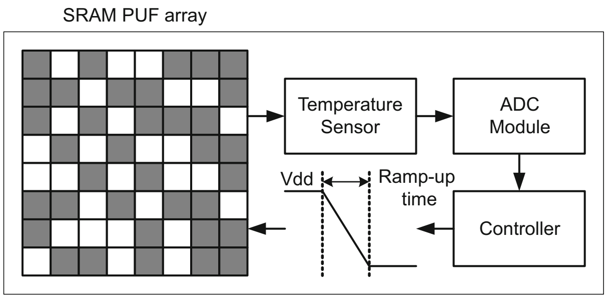


Fig. 5 Adapting the ramp-up time of supply voltage to improve the reliability of SRAM PUF

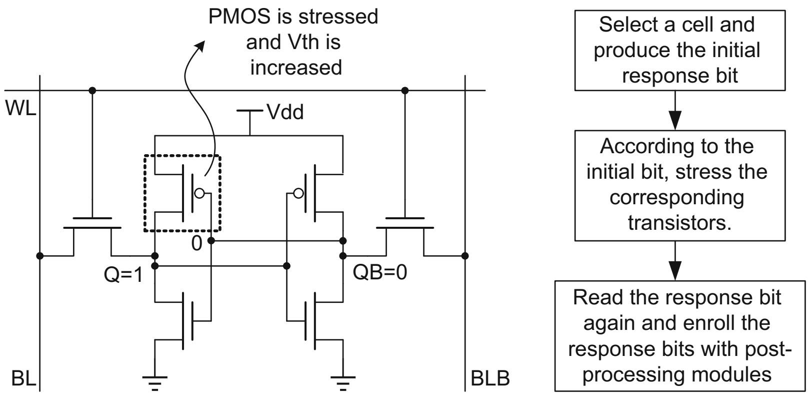

PMOS is stressed and Vth is increased
WL
Vdd
0
Q=1
QB=0
BL
BLB
Select a cell and produce the initial response bit
According to the initial bit, stress the corresponding transistors.
Read the response bit again and enroll the response bits with post-processing modules

Fig. 6 (*Left*) Aging effects to skew the threshold voltages of transistors in the SRAM-based PUF cell. (*Right*) Enrollment of SRAM-based PUF with accelerated aging procedure

aggravated by the variations of environmental conditions such as the change of ambient temperature and fluctuation of supply voltage.

One approach to improving the reliability of memory-based PUFs is to adapt the ramping time of the power-up process to generate random bits, [see Fig. 5]. The experimental results in [8, 48, 76] show that ramping the power-up time helps to reduce the BER by $\sim 71\,\%$ at room temperature. The simulation results in [15] also demonstrate that adapting the ramp-up time reduces the original BER for different types of memory-based PUFs effectively at different temperatures.

Another approach proposed in [8] used aging effects to improve the reliability of bi-stable PUFs [see Fig. 6]. It deliberately skews the characteristics of the two cross-coupled inverter structure to its preferred data bit after the chip is manufactured. Since aging effects (negative bias temperature instability (NBTI), hot carrier injection (HCI), etc.) can be accelerated if the transistors in the inverters are properly stressed, the characteristics of the two inverters will be biased towards the preferred data bit to make the output state more distinguishable. An additional procedure is required to orchestrate this reinforcement after the chip is fabricated

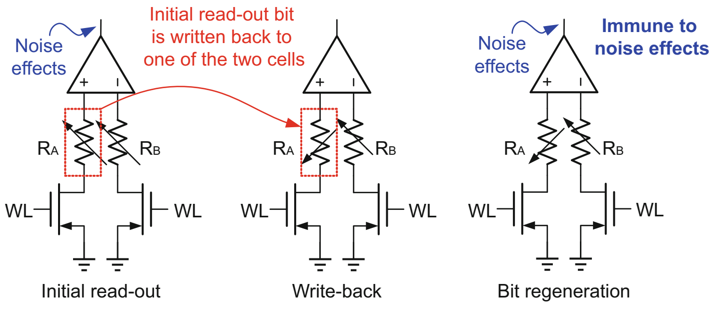


Fig. 7 Initial read-out response bit when two resistive memory cells in identical state are unstable in the presence of noise effects. Write-back operation sets one of the cells to an opposite state with respect to the other. The regenerated response bit from the cells in complementary states is more stable

and before it is used. A similar method to enhance the reliability of SRAM-based PUF is also proposed in [23]. The difference is that the later method can also be used for optimizing the entropy of the PUF cells. The idea of using HCI effects to age the PUF cell is developed in [7]. The advantage of using HCI is the shorter time required to stress the devices to obtain an acceptable distinguishability. Furthermore, the aging procedure is autonomously controlled by an internal circuit so that no response bits generated from the PUF will be leaked to the outside world.

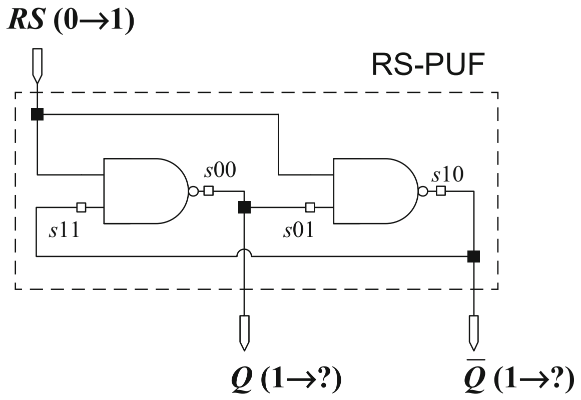
Without relying on the aging effects as well as the testing procedure, an automatic write-back scheme is used directly in the non-volatile memory (NVM) based PUF to improve the reliability of bit generation [101]. To illustrate this idea, an STT-MRAM based PUF was designed, whose initially generated bit can be written back to the cell upon completion of the read operation. The response bit can be reliably regenerated from the cell. This scheme is depicted in Fig. 7. Since the two NVM cells selected by the challenge to generate a response bit are in complementary state after a write-back operation, the BER of response bit regeneration from the cell will be very low, which is in the order of $\approx 10^{-6}$ according to the analysis in [101].

3.5 Hybrid PUF Architectures

To improve the quality of earlier discussed classical PUFs, hybrid combinations of PUFs are used [33, 92].

Figure 8 shows the R-S latch based on the main building block for SRAM based PUF. If the R and S inputs are not tied together as in Fig. 8, the inputs, S and R, can take any of the four possible combinations, namely $RS \in \{00, 01, 10, 11\}$. For normal operation, to obtain a predicted state of this latch, the input transition from $SR = 00$ to $SR = 11$ is forbidden. In this case the R-S latch will converge quickly back to one of the two stable states, which is either 1 0 or 0 1 , with a certain

Fig. 8 RS latch based PUF



preference depending on the manufacturing mismatch between the two symmetric half circuits consisting of gates and interconnecting wires. This forbidden transition can be regarded as the power-up simulation of SRAM-based cell and used as R-S latch based PUF.

After applying consecutive values of 0 and 1 in a row to the circuit input shown in Fig. 8, the stable state of Q will be random and unpredictable. This state depends on the manufacture process variations. When this circuit is replicated, manufacturing variations cause appreciable differences in the circuit delays. Across a die, device delays vary due to mask variations—this is sometimes called the system component of delay variation. There are also random variations in dies across a wafer, and from wafer to wafer due to, for instance, temperature and pressure variations during various manufacturing steps. The magnitude of delay variation due to this random component can be 5 % or more for metal wires, and is sufficiently higher for devices (gates).

Usually the device delay is evaluated as the propagation delay. The propagation delay of each gate is a function which is the product of the on resistance of the gate and the parasitic output capacitance. The equivalent gate resistance and the output capacitance are both related to the transistor dimension and that is why the delay is strongly dependent on the manufacture process variation. To increase the randomness of the variations of timing delay, the hybrid integration of two PUFs is proposed. This solution combines two approaches, namely the arbiter PUF and the R-S latch PUF, as shown in Fig. 9.

The key idea behind the proposed hybrid arbiter and RS PUF is to increase the path delays between the output of one gate of R-S latch and the input of another (see Fig. 9). The lengths of these paths are extended by n MUXs stages, which cause the delay to be strongly dependent on not only the manufacturing process variations of gates and their propagation delay, but also on the value of the applied challenge C. Depending on C, the response value will be obtained at the output of the last stage of MUXs.

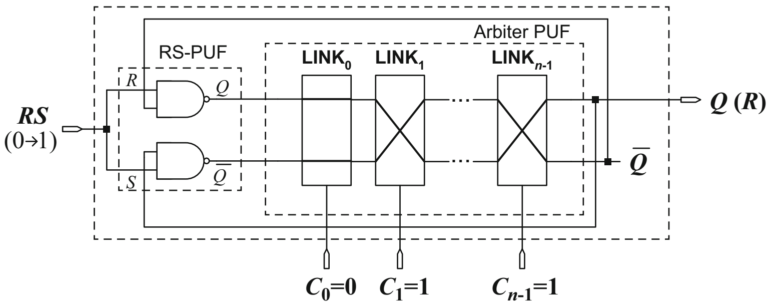


Fig. 9 Hybrid arbiter and RS latch PUF

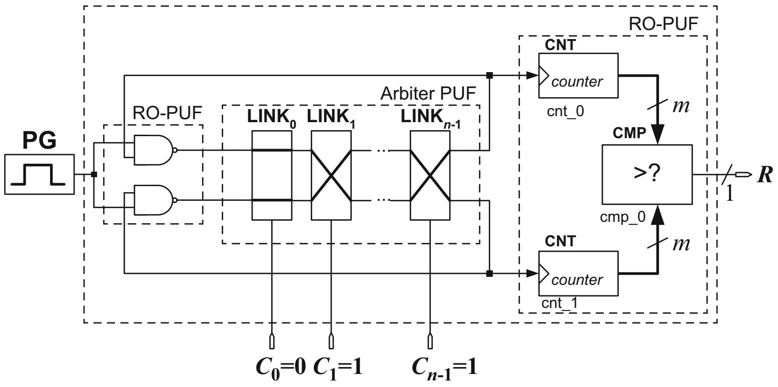


Fig. 10 Ring oscillator and arbiter PUF

Another example of hybrid PUF solution is the combination of RO PUF and arbiter PUF, as shown in Fig. 10.

4 Practical Examples of PUF Implementation

This section presents some examples of PUF implementation for identification, authentication and secret key generation.

4.1 PUF-Based Crypto-Processor for the RFID Systems

Radio Frequency IDentification (RFID) systems are widely used in retailing, assembly lines, pharmaceuticals, transport, access control systems, libraries, custom, agriculture, livestock and pets, etc., to transfer data wirelessly for the purpose

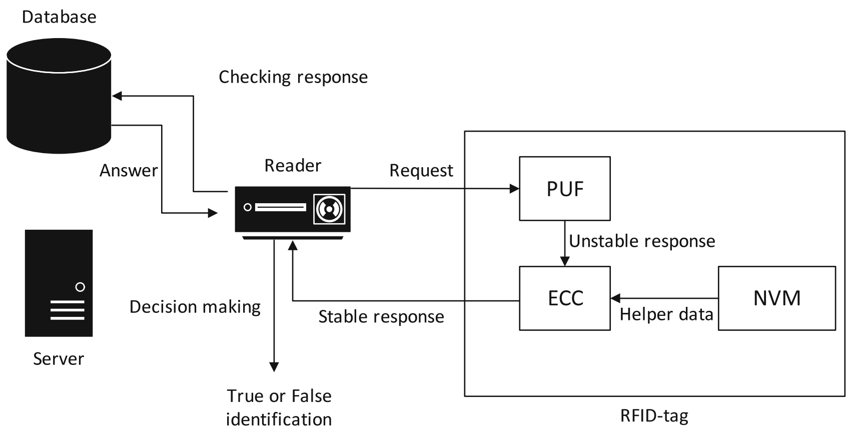


Fig. 11 PUF-based RFID system structure

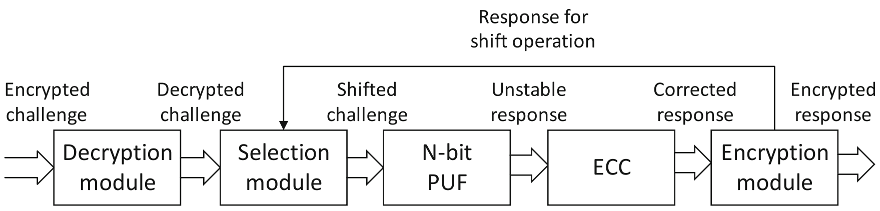


Fig. 12 PUF-based encryption processor

of automatically identifying and tracking the tags attached to objects. Cost and privacy protection are the two major concerns in using RFID-based identification and tracking systems. By using PUF as the data storage and processing unit, unauthorized reading of sensitive personally-linked information can be prevented with lower hardware cost.

A simple RFID system comprises at least a reader and a transponder (RFID tag) as shown in Fig. 11. To identify the RFID tag with the reader, some challenge (or challenges) to the tag is sent. The PUF, upon receiving the challenge, generate a response. The unstable response is corrected by the ECC-block. The post-processed stable response is sent back to the RFID reader for checking against the local database or a database located on a remote server. If the response of RFID-tag is found in the database, the RFID tag is authenticated. This decision making process is both fast and secure because PUF is a low-cost hardware entity and its responses are unique and unpredictable.

The structure of a PUF-based crypto-processor for RFID applications [14] is depicted in Fig. 12.

Decryption module receives the encrypted challenge, the length of the challenge and the random number from the server to decrypt the challenge. The encryption method depends on the instruction code, which is based on XORing some bits of

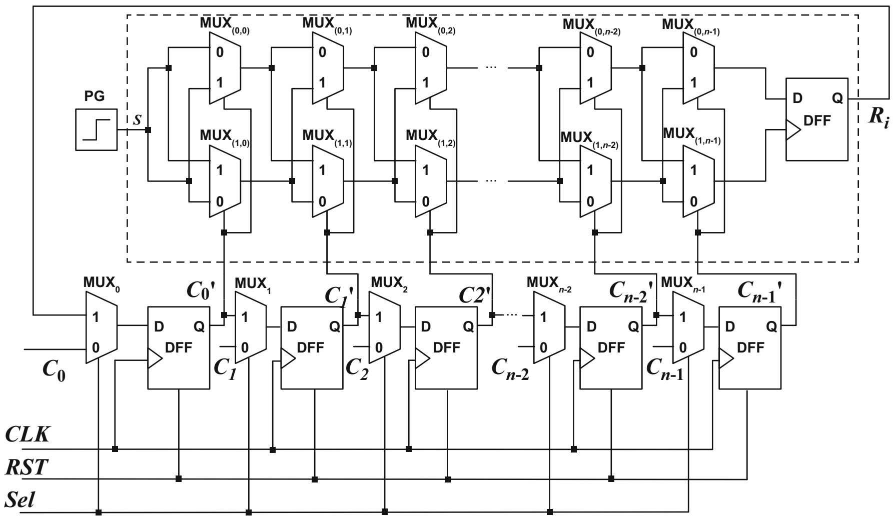


Fig. 13 An N-bit arbiter based PUF

the challenge and the random number generated from the server. The length of an instruction code is $N/2$ bits. Therefore $2^{N/2}$ encryption methods are possible. These methods have to be simple because the communication between the RFID-tag and the reader has to be as fast as possible.

The N**-bit PUF** is designed based on the classic arbiter PUF. To get an N-bit response from a 1-bit response, N-bit shift register is proposed for the challenge modification. Every response bit is obtained from the shift register by feeding back the previous response. N such iterations are used to produce an N-bit unique PUF response. The structure of the arbiter PUF is depicted in Fig. 13.

The **ECC-block** corrects the noisy PUF response to achieve a strong authentication. The correction code used was not mentioned. As the responses generated by the arbiter PUF are very stable, the use of Hamming or repetition codes is adequate to eliminate the noise.

Finally, two tasks are performed by the **Encryption module**, which are encrypting the corrected PUF response and providing the feedback information for the N-bit PUF iterations.

This authentication protocol meets the cost and speed criteria for RFID applications. The method is resistant against various kinds of attacks, such as physical, modeling, spoofing and location tracking. The shortcoming of this work [14] is that the impact due to the variations of environmental conditions was not investigated.

The first commercially available RFID tag with embedded PUF was developed by Verayo Inc. [86]. The RFID system uses authentication protocol and its PUF-responses are checked by Hamming distances. The general structure of the system is similar to the one mentioned above (see Fig. 11) without the ECC block. The details of the actual implementation is proprietary, but the test data are provided by [36]. The test results confirmed its reliability and uniqueness. The developed RFID tag shows excellent performance under a wide range of temperature (from −45 to 95 °C). In addition, Verayo Inc. also uses PUF for FPGA IP-protection, anti-counterfeiting, secure IDs and access, and several other security solutions.

4.2 FPGA IP-Protection with PUF

Digital Watermarking is widely used for the copyright protection of digital artifacts. Its salient feature, as suggested in 1992 by Andrew Tirkel and Charles Osborne [81], is "The watermark is capable of carrying such information as authentication or authorization codes, or even a legend essential for image interpretation". Today, digital watermarking is used for more than the copyright protection of images. The watermark is usually a secret that is used to prove the ownership or authorship of a digital IP and is stealthily embedded into any form of IPs to be protected against piracy or misappropriation. Digital fingerprint is an extension of digital watermark. It carries both the information about the IP-owner and the legal user of the IP. From the application point of view, the PUF responses can be considered as the fingerprint of a digital device. Extensive research works have been devoted to the use of PUF as device fingerprints [21, 29, 30, 57]. Due to the flourishing FPGA IP market and the vulnerability of FPGA bitstreams to illegal copying and cloning, the bitstream are usually encrypted but the storage of the encryption key poses additional costs and security concern. Unique fingerprints generated from an intrinsic PUF without overhead by exploiting the internal SRAMs of FPGA chips have been proposed as an alternative solution. Each SRAM bit-cell is essentially an RS-latch implemented with six transistors arranged as two cross-coupled inverters and two access switches. When the SRAM cell is off, the R and S inputs are both set, which gives rise to an unstable state. When the voltage supply to the cell is turned on, the two unstable logical states must change values and the final stable logic value read out from the output Q (or $\overline{Q}$) will be unpredictable [29]. The variations in the relative threshold voltages of the transistors making up each cell cause the bit to flip randomly after the initial state. All memory cells can thus be divided into three classes: zero-skewed (the values after power-up tend to zeros), one-skewed (the values tend to ones), mix-skewed (the values are almost uniformly distributed). Experiments conducted on three identical Virtex II Pro boards [47] show that greater than 90 % of the 4096 SRAM cells are zero-skewed, less than 10 % are one-skewed and the remaining less than 1 % are mix-skewed.

A typical fingerprint extraction algorithm consists of the following two main steps:

– **Preselection.** The digital device is powered-up n times. The most reliable bits are chosen to be the fingerprint bits. The reliability r for bit b_i can be calculated by Eq. (14), where m is the larger value of the number of occurrences of zeros and of ones. For example, the bit is reliable if $r(b_i) \geq 95\,\%$.

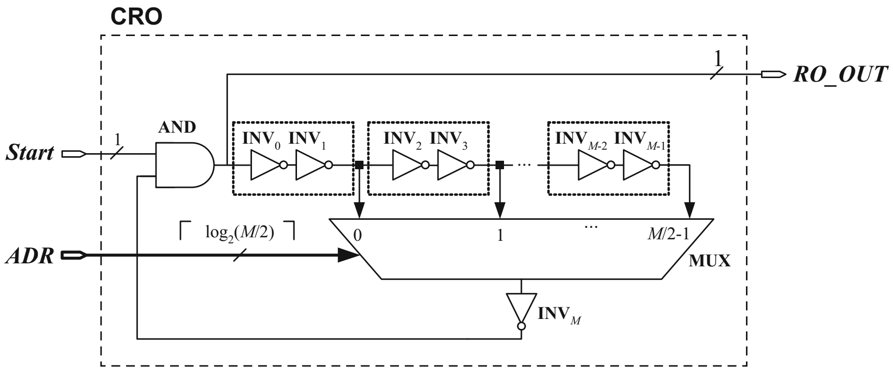


Fig. 14 Configurable ring-oscillator structure

$$r(b_i) = \frac{m(b_i)}{n(b_i)} \cdot 100\,\% \tag{14}$$

– **Helper data collection.** The bits with the high reliability may be corrected by simple and low hardware cost ECC. Therefore, helper data for the chosen ECC algorithm is collected to perfectly reproduce the response bits. If identification is provided by the Hamming distance comparison, then ECC is not necessary.

According to [29], only 19 % of all SRAM cells are selected as the fingerprint bits. Thus, SRAM-based approach is highly redundant for fingerprint extraction and unreliable by both Hamming distance based and exact matching. Another problem is that there are too few unique chips per fingerprint size (64-bit identifier for the identification of only about 4,000 devices).

Another approach is based on the modified Ring Oscillator PUF (RO-PUF) [32]. A configurable ring-oscillator structure is depicted in Fig. 14. Two parameters, *START* pulse duration and address *ADR* to the multiplexer data select lines are associated with the circuit implementation of this PUF. The registered pulse count on the output port $ROOUT$ is the response of the PUF.

Fingerprint extraction experiments are conducted on two Xilinx SPARTAN-3E (XC3s500e-5FG320) FPGA prototype chips, B_0 and B_1. The digital device (see Fig. 15) for pulse registration consists of a start pulse generator SPG, a configurable ring-oscillator CRO, a pulse counter CNT and a LED-indicator controller SLC. The 50 MHz system clock generator CLKG, the hardware buttons BNT1 (asynchronous initialization signal), BTN2 (result display controller) and three switches SW(2:0) are included in the digital system Digilent Nexys-2 to control the device's operation.

The first stage of the experiment is digital pulse monitoring. The graph in Fig. 16 plots the difference between the pulse counts $N_R(k)$ of the two boards versus the scaling coefficient of enabling signal delay time (D_S), where $k = 2^i$

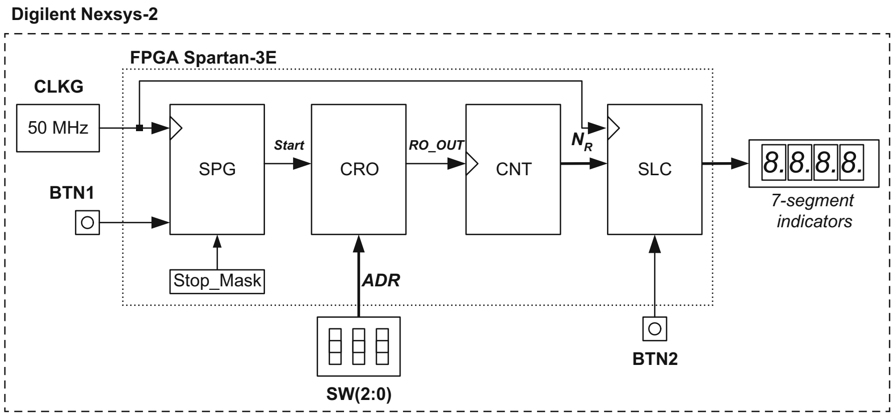


Fig. 15 Digilent Nexsys-2 digital device structure

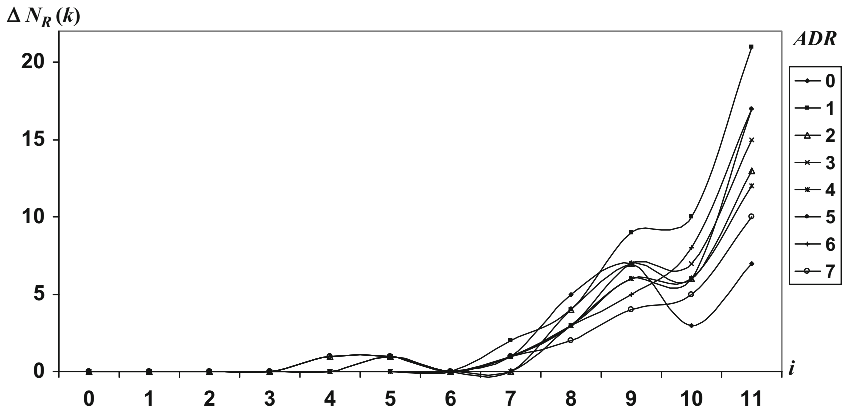


Fig. 16 Difference between the pulse counts of the two broads for different scaling coefficient $k = 2^i$ and all possible values of *ADR*

for $i = 0, \ldots, 10$. $\Delta N_R(k)$ becomes non-zero when $k = 2^4$ and increases linearly with k for $k \geq 2^7$ for all addresses (*ADR* in Fig. 14) of the RO pair.

Experimental data for $k = 2^{10}$ ($kD_s = 2^{10} \cdot 20 = 20,480$ ns) are shown in Table 4, where $N_R^{B_0}(k)$ and $N_R^{B_1}(k)$ are the numbers of registered pulses expressed in hexadecimal number format for the digital systems B_0 and B_1, respectively. The arithmetic difference between the two pulse counts are listed in the row labeled $D_\Delta(k)$ and the Hamming distance in the row labeled $H_\Delta(k)$. Their average values for this experiment are $\overline{D_\Delta(k)} = 7$ and $\overline{H_\Delta(k)} = 2$, respectively.

The second stage of experiment is performed by running the digital devices B_0 and B_1 100 times with $kD_S = 20480ns$ for all values of *ADR* to monitor the pulse number deviations. The most frequently appeared number of registered pulses ($N_R^{B_i}(k)$) in the 100 trials is recorded. The absolute values of deviation for all values

Table 4 Experimental data for $kD_s = 20,480$ ns

	ADR							
Parameters	0	1	2	3	4	5	6	7
$N_R^{B_0}(k)$	7FC	671	55D	4BB	40F	396	33D	2F2
$N_R^{B_1}(k)$	7F8	666	557	4B3	408	38F	334	2ED
$D_\Delta(k)$	4	11	6	8	7	7	9	5
$H_\Delta(k)$	1	4	2	1	3	3	2	5

Table 5 Probabilities of occurrences of symbol "1"

	Binary bit positions										
Parameters	10	9	8	7	6	5	4	3	2	1	0
$N_R^{B_0}(k)$	1.0	0.0	0.0	0.0	0.0	0.0	0.9	0.1	0.1	0.1	0.15
$N_R^{B_1}(k)$	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0	0.07	0.92

Table 6 Binary identifiers of digital systems B_0 and B_1

	Binary bit positions										
Parameters	10	9	8	7	6	5	4	3	2	1	0
$ID(B_0)$	1	0	0	0	0	0	**1**	**0**	0	0	**0**
$ID(B_1)$	1	0	0	0	0	0	**0**	**1**	0	0	**1**

of *ADR* is less than 2. For example, for $ADR = 4$, the pulse generator implemented in B_0 produces the following $N_R(k)$ values in hexadecimal representation: 410 (85 %), 40F (10 %), 411 (5 %) and that implemented in B_1 produces 409 (92 %), 40A (7 %), 408 (1 %). The probabilities of occurrences of symbol "1" in every output bit positions in this experiment are given in Table 5.

To extract the unique chip fingerprints, the above probabilities are used. The resulting chip IDs are computed by majority principle and given in Table 6. The different bit values between the two identifiers are printed in bold. The experiment results show that the Hamming distance between two 11-bit fingerprints (for scaling coefficient $k = 2^{11}$) is 3, which means that about 400 XC3s500e-5FG320 FPGA chips can be strongly identified. Increasing the value of k will enable more devices to be identified.

4.3 PUF-Based True Random Number Generator (TRNG)

Random number sequences are widely used in cryptography, modeling, gaming industry, random sampling, decision making processes, art and etc. [11]. For cryptography, the random numbers have to be unpredictable to avoid any security vulnerabilities. Typically, a TRNG has three major components.

– **Entropy Source.** To create true random number sequence, a source of randomness based on some physical process (atmospheric noise, thermal noise,

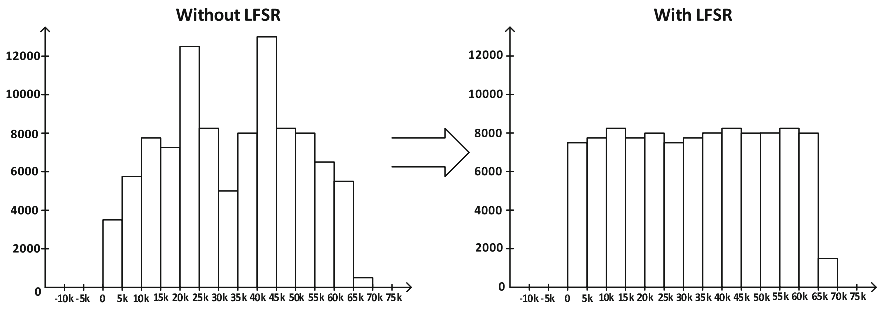


Fig. 17 Frequency distribution of random sequences of RO-PUF before and after augmented by LFSR

electromagnetic and quantum phenomena, etc. [65]) is needed. As PUF responses are generated based on the intrinsic chip process variations, PUF can be used as an entropy source. FPGA based implementations of TRNG use different sources like ring-oscillator PUF (RO-PUF) [3, 27, 38, 50, 80, 82], SRAM-PUF [29], arbiter PUF [75], automatic phase locked loop (PLL) [16, 17, 85], delay locked loop (DLL) [42], thermal noise [64] and others. PUFs are high-quality and high-speed entropy sources which are easy to implement and have low hardware cost.

- **Compression Scheme.** A single source of randomness may yield low quality random sequence. The quality of a random sequence is assessed by NIST [74] and Diehard [53] statistical test results. The following approaches are used to enhance the random sequence quality: XOR tree [17, 27, 50], linear feedback shift register (LFSR) [75], sampler [38, 42], von Neumann corrector [16] and alternative step generator for oscillator-phase-noise decorrelation [82]. Our investigation [98] shows that augmenting the entropy source (RO-PUF) by LFSR and XOR-tree makes the random sequence more uniformly distributed (see Fig. 17).
- **Random Number Register.** After post-processing (compression), the random number will be stored in register. The stored information may be transmitted directly to the end-user or used as cryptographic nonces in authentication protocol.

Experiments were conducted for different PUF based TRNGs implemented on Xilinx SPARTAN-3E (XC3s500e-5FG320) FPGA chips mounted on two Digilent Nexys-2 prototype boards, B_0 and B_1. Xilinx ISE WebPack [89] was used for the VHDL project design and the FPGA was configured with Digilent Adept software [18]. The data analysis was conducted with NIST and Diehard statistical test packages and STATISTICA line of software [77].

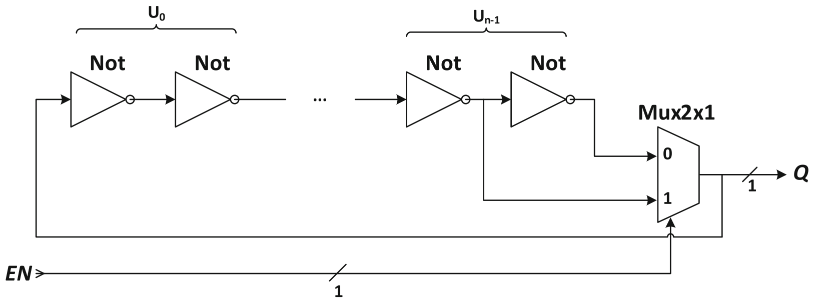


Fig. 18 Modified RO-PUF structure

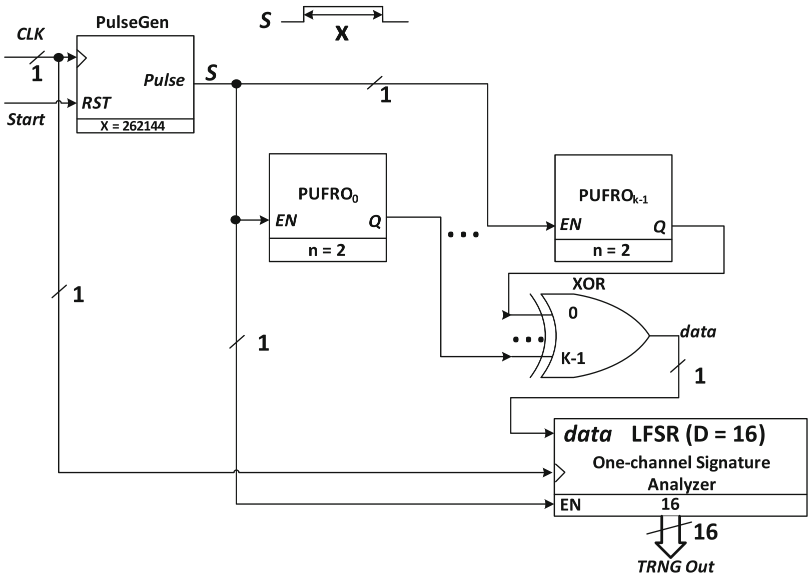


Fig. 19 RO-PUF based TRNG

4.3.1 RO-PUF Based TRNG

Figure 18 shows the modified RO-PUF structure used for the construction of entropy source. It is built up of $2n+1$ inverters with a feedback path. The number of inverters should be odd to create a meander output signal. Instead of counters in the classical structure, multiplexer is added to provide two modes of operations: SRAM-PUF (EN = “0”) and RO-PUF (EN = “1”).

In RO-PUF mode, the output frequency F_Q will be unique, unclonable and unpredictable for each chip. Therefore, this mode can be used to generate random numbers. However, as mentioned earlier, this randomness has low quality. To overcome this problem, the TRNG circuit shown in Fig. 19 is proposed.

Table 7 Sample Pirson correlation coefficients for RO-PUF based TRNG

	B_0 first launch	B_0 second launch	B_1 first launch	B_1 second launch
B_0 first launch	1.000000	0.003840	−0.002286	−0.001483
B_0 second launch	0.003840	1.000000	0.003972	−0.002834
B_1 first launch	−0.002286	0.003972	1.000000	−0.004674
B_1 second launch	−0.001483	−0.002834	−0.004674	1.000000

Table 8 RO-PUF based TRNG hardware costs

Resource	Used	Available	% of used
Slice flip flops	273	9,312	3
LUTs	1,883	9,312	20

This circuit consists of four blocks:

– **Single Pulse Generator.** This block generates single pulse with pulse width X. This pulse defines the RO-PUF's behavior.
– **PUF-chain.** k RO-PUF entities are chained to generate the initial random number sequence.
– **XOR-tree.** A k-input XOR-gate is used to obtain one output bit from k PUF response bits.
– **LFSR.** This block is used to generate pseudo random number sequences but with true random seed. A D-digit true random number is produced by this block. It also represents a one-channel signature analyzer.

Thus, the TRNG is configured by the tuple (n, X, k, D). The configuration ($n = 2$, $X = 262144$, $k = 256$, $D = 2$) is used to generate $1.6 \cdot 10^8$ random bits. More than 90 % of the samples have successfully passed all NIST tests for both chips. The Diehard tests results also confirm the high random sequence quality.

The correlation between four random sequences generated by B_0 and B_1 was also investigated. The sample Pirson correlation coefficients are shown in Table 7. Our experiment results show that the sample Pirson correlation coefficients are small (the absolute value is less than 0.005). Our statistical hypothesis test shows no Pirson correlation between different sequences at a significance level of $\alpha = 0.05$. In conclusion, our test results confirm that this TRNG can produce unclonable, unpredictable, non-reproducible and uncorrelated true random number sequences.

The hardware resources required to implement this TRNG with $n = 2$, $X = 262,144$, $k = 256$ and $D = 2$ on FPGA are shown in Table 8. Comparing with other RO-PUF based TRNGs, the proposed implementation uses four times less number of flip-flops but double the LUTs [50]. Other TRNGs like [82] would require twice as much hardware costs of our implementation.

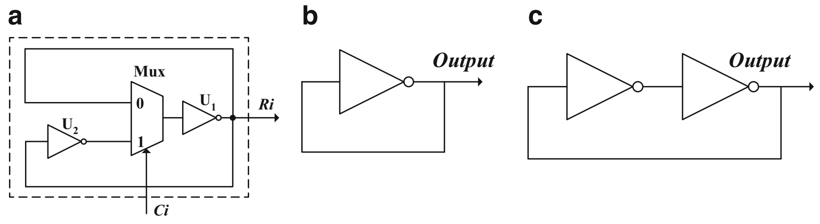


Fig. 20 Dual-mode hybrid PUF: (**a**) complete structure, (**b**) RO-PUF mode, (**c**) SRAM-PUF mode

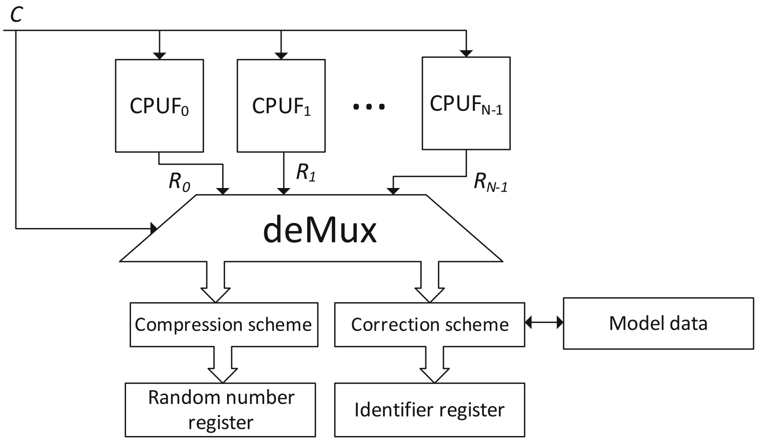


Fig. 21 General structure of SRAM-based TRNG and unique ID generator

4.3.2 SRAM-PUF Based TRNG

The hardware overhead of the TRNG can be reduced by creating entropy source from hybrid PUF [95]. A dual-mode hybrid PUF is implemented by two invertors and a multiplexer, as shown in Fig. 20a. Similar to the modified RO-PUF of Fig. 18, this circuit can also operate in two modes: RO-PUF (*Challenge* = "0") (Fig. 20b) and SRAM-PUF (*Challenge* = "1") (Fig. 20c).

In the SRAM-PUF mode, the SRAM-cell behavior is emulated by the bistable element (i.e., couple of invertors). This device can be used for unique chip ID generation and to create random sequences from the IDs. The general structure is illustrated in Fig. 21.

The circuit needs to be powered up to generate the ID. Experimental results show that the ID repetition probability is 0.02 and about 81.25 % of the ID bits are stable. Thus, the power-up states of the SRAM-PUF can be used as chip ID with preselection and error correction. There is no need for ID stabilization if the main objective is to build a TRNG.

Figure 22 provides a visual interpretation of a 64-bit ID. For each cell c_i, a fully black cell represents $p(c_i) = 1.0$, a fully white cell represents $p(c_i) = 0.0$ and a

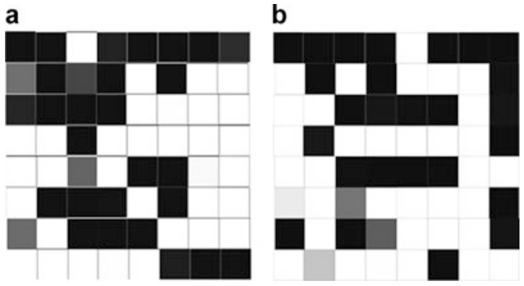


Fig. 22 The probabilities of state one output for the SRAM-cells in **(a)** chip B_0 and **(b)** chip B_1

Table 9 Distance metrics between IDs of B_0 and B_1

Distance metric	Min distance	Max distance	Avg distance	Distance between chip ID
Euclidean distance	4.58257	5.29150	4.90749	5.19615
Hamming distance	21.0000	28.0000	24.11000	27.00000
Minkowski distance ($p = 3$)	7.00000	9.33333	8.03667	9.00000
Correlation distance	0.66386	0.89579	0.76400	0.85820
Cosine distance	0.27994	0.39975	0.33585	0.49015
Correlation distance	0.32812	0.43750	0.37672	0.42188

Table 10 Average Hamming distance between projects, P_0, P_1 and P_2, loaded in B_0

	Project P_0	Project P_1	Project P_2
Project P_0	0.0	27.05	33.26
Project P_1	27.05	0.0	29.25
Project P_2	33.26	29.25	0.0

Gray color cell represents $0.0 < p(c_i) < 1.0$, where $p(c_i)$ is the probability of logic one output for each SRAM-cell c_i in n power-up experiments. More black cells imply larger $p(c_i)$ value.

To calculate the chip ID, majority principle was used. As shown in Fig. 22, the probability distribution of the SRAM cells for each chip is different. This is also evident from the different distance metrics between the IDs of the prototype boards, B_0 and B_1, obtained in Table 9. The relatively large Hamming distances are good evidences that the 64-bit chip IDs generated by this PUF are distinctive.

Different FPGA projects result in different circuit topologies. The average Hamming distances of three different projects loaded onto the same prototype board are shown in Table 10. It can be seen that the IDs generated by the SRAM PUF differ not only between chips, but also vary with the projects loaded in a chip.

Adaptive signature analyzer (ASA) [35] were proposed as a compression scheme to generate unique, unclonable and unpredictable random number sequences from the chip ID. The 64-bit chip ID is compressed into a 6-bit number S by:

$$S = \oplus_{i=0}^{N-1} A_i, \forall ID[A_i] = 1 \tag{15}$$

Table 11 Hardware costs of SRAM-PUF based TRNG

Resource	Used	Available	% of used
Slice flip flops	15	9,312	0.2
LUTs	710	9,312	7.6

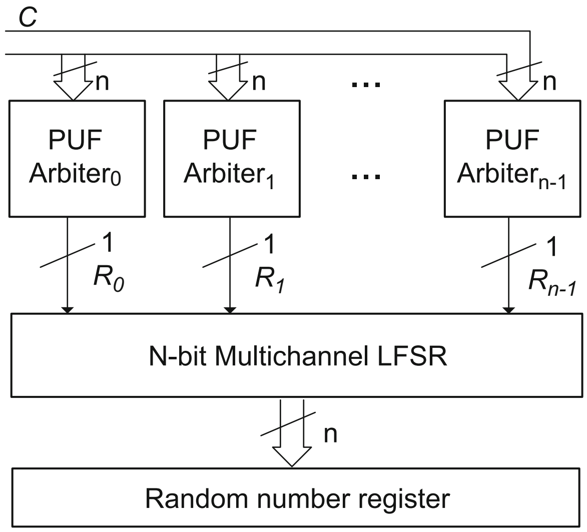


Fig. 23 Arbiter PUF based TRNG

where A_i is the SRAM cell address (a number in [0, 63]), N is the number of SRAM cells and $ID[i]$ is the bit value of the ID stored in address i.

To test the randomness, a $6{\cdot}10^7$-bit sequence is generated by the proposed TRNG. This bit-sequence was tested by NIST and Diehard packages. Pirson correlation test was also conducted. The test results proved that the generated true random number sequences are of high quality.

Thus, this approach accomplishes two complementary tasks: unique chip identification and true random number generation. Similar results were obtained by Holcomb et al. [29] but our results are better in terms of resource utilization, especially in the number of flip-flops used, and bit stability. The hardware costs, excluding the cost of the signature analyzer, for the generation of a 256-bit ID are shown in Table 11.

4.3.3 Arbiter PUF Based TRNG

Instability of classic arbiter PUF responses is a well-known problem [26]. The use of N parallel arbiter PUFs to whiten the responses was proposed in [97]. Multichannel LFSR [4] was used to compress the generated bit-sequence. The general TRNG structure is shown in Fig. 23.

To generate random number sequences, all possible n-bit challenges are applied to the arbiter PUF's inputs. The PUF responses are fed to the multichannel LFSR to obtain a uniform distribution. Similar experiments as before are conducted and random sequences of high quality are also confirmed by statistical testing (NIST,

Table 12 Hardware costs of arbiter PUF based TRNG

Resource	Used	Available	% of used
Slice flip flops	20	9,312	0.2
LUTs	163	9,312	1.8

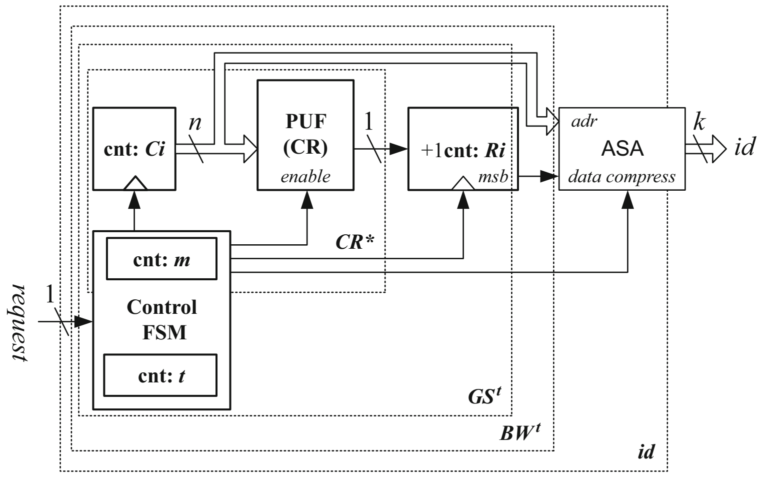


Fig. 24 Arbiter PUF based identification device structure

Diehard, statistical hypothesis). The hardware costs, excluding multichannel LFSR, are shown in Table 12. The most prominent advantage of this implementation is the savings of more than four times of hardware resources over other TRNGs. Its main drawback is the very slow generation of random sequences (about 10 Kb per second by USB interface).

Another approach to use arbiter PUF for unique chip ID generation is proposed in [34]. The identification device structure is depicted in Fig. 24. The CR^* block generates unstable challenge-response pairs. The GS^t block generates the bit probabilities while the BW^t block stabilizes the responses. Finally, the *id* block generates the identifier.

The arbiter PUF challenge-response pairs were formed by challenge repetition ($t = 15$ times). Experiment results indicated that the responses were unstable. To obtain stable response, the t responses are transformed into a binary number by majority principle. As it is inconvenient to use a n = 256-bit stable response to identify an FPGA chip, the response is compressed by an ASA. According to our experimental results, the two FPGA chips B_0 and B_1 were found to have different stable 8-bit IDs ($ID(B_0)$ = "01111000" and $ID(B_1)$ = "11101100").

Our investigation of the above mentioned FPGA PUF implementations show that PUF can be used as a source of unique chip fingerprint (identifier) and true random number sequences. To accomplish these two opposite tasks, the PUF responses will have to be processed in different ways. From the experiments conducted on the physical prototypes, the investigated PUFs are very attractive entropy sources with high security, low hardware resources utilization, response uniqueness, unpredictability and unclonability [96].

5 Emerging Types of PUFs

Besides conventional types of PUF, there is a group of emerging PUF instances featuring new properties and/or retrofitting applications.

5.1 Reconfigurable PUF

Reconfigurable PUF (rPUF) is a kind of PUF whose challenge-response mapping characteristics can be altered given a different **configuration state** (denoted as s hereafter).

The original conception of rPUF was proposed by Kursawe [41], though there are some other PUF instances that have similar feature but different notations. Le et al. [102] categorize rPUF into two groups, namely **Logically rPUF** (L-rPUF) and **Physically rPUF** (P-rPUF). By definition, the L-rPUF relies on logical interfaces (often called **control logic**) to obfuscate the challenge-response pairs of its internal PUF according to the configuration state. P-rPUF exploits the intrinsically alterable properties of the PUF device itself to realize the reconfiguration. This physical property alteration is triggered by stimulating the PUF with an external effect. The operation mechanisms of L-rPUF and P-rPUF are shown in Fig. 25.

In [41], the authors introduce an implementation of P-rPUF that exploits the sensitivity to optical stimulation of certain material to achieve physical reconfiguration. Each time the material is exposed under a laser beam, its physical characteristics will be disordered. The response bits extracted from the PUF change after this process.

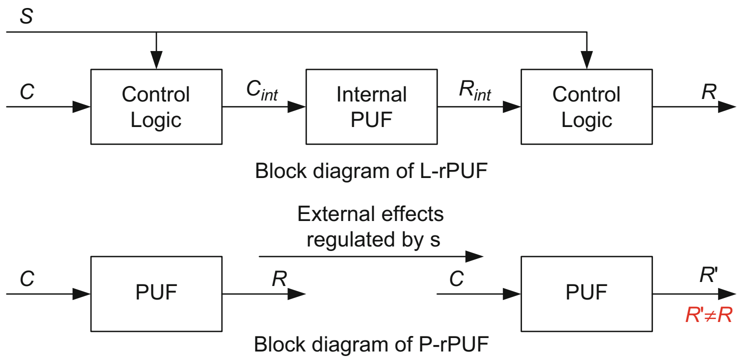


Fig. 25 Block diagrams of L-rPUF (*top*) and P-rPUF (*bottom*)

Another P-rPUF instance proposed in [41, 100] exploits the measurable resistance of phase change memory (PCM) cells to produce the response bits. By re-programming the cell with new electrical pulses, resistance values of the PCM cells can be changed, which in turn changes the response bits. Detailed analysis and experimental demonstration of PCM-based rPUF are further presented in [102]. A novel design of PCM-based rPUF that has enhanced security against prediction attacks was also developed. Instead of relying on the measurable resistance, the number of electrical pulses that converges a cell resistance to a pre-determined value is used as a response.

Rhumair et al. [73] defined a PUF primitive called "Erasable" PUF, which is essentially a P-rPUF. It is constructed based on resistive memory structured as a cross-bar array. Whenever a voltage across the memory cell is higher than the threshold, the I-V characteristics of the cell will be altered. Therefore, the measured electrical characteristic that is digitized as response bit is refreshed.

Device aging effects were also applied to change the device physics so as to alter the challenge-response mapping of a PUF. Meguerdichian and Potkonjak [54] proposed the aging PUF to match the challenge-response characteristics of two PUFs to establish an authentication protocol between two parties.

On the other hand, L-rPUF can be dated back to [25] where the "Controlled" PUF was proposed. It is basically a hash function that interfaces the input challenge and the output response. Due to the diffusion effect of the hash function, the real challenge and response of the PUF are hidden. The controlled PUF can be used to thwart certain man-in-the-middle attack. A generalized version of the controlled PUF is the "Logically Recyclable" PUF [19, 37]. It also uses control logic to interface the challenge and response, but the interface is implemented as block cipher whose state is kept secret. By initializing the PUF with different configuration states, the challenge and response are diffused by the cipher module in a different way.

rPUF has the following advantages over the conventional PUFs for use in security-critical systems and highly information-sensitive applications.

- Its challenge-response behavior can be dynamically updated.
- The space of possible responses of PUF is expanded.

In [41], rPUF is proposed to use for the protection of persistent storage of security module. Owing to the reconfigurability of rPUF derived keys, malicious update of the keys such as invasive setting of the internal states of the storage, can be detected. In [102], the advantage of rPUF is discussed for its enhanced resilience against prediction attack. The authors of [73] introduce a key-exchange scheme based on the Erasable PUF to resolve a security issue in a conventional protocol.

Apart from using rPUF to improve the security level of the PUF primitive itself or the protocol constructed from it, rPUFs are also proposed to improve the reliability of certain PUF instances. The authors of [49, 90, 94] independently proposed the use of configurable structure on FPGA platform to design ring-oscillator PUF that has better quality. The basic idea is to choose fundamental elements through a multiplexer to achieve a ring-oscillator PUF that has the best performance in terms of reliability and uniqueness.

Logically recyclable PUF introduced in [37] makes PUF reusable in certain applications such as recyclable RFID-based authentication token, air flight luggage tags, secure update of secret keys, prevention of malicious downgrading of software version, etc.

5.2 SIMPL/Public PUFs

Simulation-Possible-but-Laborious (SIMPL) systems [12, 13, 68–70, 72] and Public PUFs (PPUFs) [6, 51, 54, 62, 66, 88] refer to a group of PUFs (or PUF systems)[1] that can be used in a way analogous to public-key cryptography. Basically, a PPUF has the following features [70]:

- PPUF is a physical system that can generate a device-specific response R when stimulated by a challenge C due to the physical disordered properties of the system;
- The challenge-response mapping behavior of the physical system is assumed to be stable over time;
- A numerical algorithm, denoted as Sim, exists to calculate R according a given C. But the speed of obtaining a response using Sim from the system is remarkably slower than directly deriving the responses from the system.

Owing to the asymmetry between physical evaluation and computer simulation, the PPUF always computes the responses to the given publicly known challenges faster than any other means. While the asymmetry of public-key cryptography stems from the use of different keys to perform the function and its inverse, PPUF relies on the heavy mismatch between the efficiency of physically synthesized and software simulated response generation to establish security protocols that can achieve similar effects. Instead of relying on the mathematical intractability of deriving the private key from the public key, the strength of PPUF protocol lies in the fact that the adversaries are unable to obtain the secret keys in time to break the cryptographic system.

As an example, assuming Alice (denoted as A) and Bob (denoted as B) are communicating based on a simple identification protocol. A physically holds the PPUF and has made the simulation method Sim of the PPUF public. A can then prove her identity to B via the protocol as shown in Fig. 26 [70].

Other protocols such as authentication, key-exchange, bit-commitment and zero-knowledge proof can also be established based on PPUFs.

There are a number of implementations of PPUF proposed in the literature. In [12, 72], SRAM based and Cellular Non-linear Network based SIMPL systems

[1]For convenience, we refer to both SIMPL system and PPUF as PPUF hereafter.

```
$B$ chooses $k$ challenges $C_1, C_2, \ldots, C_k$;
for $i = 1 \to k$ do
    $B$ sends $C_i$ to $A$;
    $A$ uses the PPUF to produce $R_i$ according to $C_i$, and send $R_i$ back to $B$;
    If $B$ does not receive $R_i$ from $A$ within a time bound, he sets a verifier $V_i$ as $\perp$;
end for
$B$ computes all $R_i$ $(1 \leq i \leq k)$ and verifies if $R_i = V_I \neq \perp$;
if This is the case then
    The identity of $A$ is verified;
else
    The identity of $A$ is denied.
end if
```

Fig. 26 Identification protocol based on PPUF

are proposed. The former design essentially exploits the reliance of SRAM write operation on the scaling of power supply voltage. If the supply voltage is over-scaled, write malfunction will occur and the data bit will fail to be written into the cell. Simulating successive write-after-read operations performed iteratively on a large array is always much less efficient compared to the realistic physical execution of the same procedure on SRAM chips. The latter design exploits the two-dimensional analog array to realize a PPUF. Since the communication with the CNN is by nature via the transmission of analog signals, it is much faster compared to software simulation. A speed gap between the physically evaluated responses using the CNN based PUF and the simulated response is big enough to construct a public-key cryptography from it.

In [6, 51], PPUFs are constructed based on a network of XOR gates. Though the delays of the XOR gates are publicly known, owing to the interleaving inputs and outputs of the gates, numerically evaluating the output with the known XOR gate delays is a much tedious job compared to obtaining the results directly from the circuit itself. The idea of PPUF was further developed in [62] as differential PPUF. This new primitive removes the reliance of the XOR-gate based PPUF on accurate clock manipulation. Instead, it uses two input signals to race against each other for the purpose of evaluating a final output. Matched PPUF is another variant following this train of thoughts [54]. It exploits aging effects of IC devices to achieve accurate match of two individual PPUF devices that cannot be matched with a third one. In this manner, two parties can establish a secure communication protocol by using their matched PPUFs, and any other malicious attempts to match the devices will be futile owing to their extremely low efficiency.

Besides conventional CMOS based implementation, PPUF can also be realized by emerging technologies. In [66], the PPUF is designed by memristors configured in a cross-bar array. By making use of the non-linearity of memristive device and the bi-directionality of signal transmission between adjacent devices, sneak path that is exponential to the size of the array can possibly be found. By randomly selecting M-omino shapes from the array to transmit signals to obtain the outputs at certain nodes, it takes much longer time to obtain the results via software simulation than physical evaluation. A similar idea was proposed in [88] as "Bidirectional Polyomino Partitioned" PPUF where either memristors or nanowires are used as media to construct a PPUF that relies on the uniqueness of the partitions of bidirectional array to achieve the asymmetry between simulation and physical evaluation.

6 Conclusion

Physical cryptography has become heated research topics in the recent years. Physical Unclonable Functions that exploit the physical disordered properties of integrated devices/systems to produce device-specific responses given certain stimulus, have been studied for a variety of security applications such as hardware identification, device authentication, secret key generation, random number sequence generation, unique serial numbering, etc. The past few years have witnessed a great many research achievements triggered by the invention of PUF including new applications of PUFs, quality enhancement techniques, and novel implementations of PUF primitives and associated protocols.

In this chapter, we reviewed the fundamentals of PUF, and then gave an extensive introduction of how PUF can be applied in the contemporary hardware-based security applications. We also presented a broad overview of the classical as well as the state-of-the-art techniques for PUF quality enhancement. We then presented the implementation methods of conventional PUFs for alternative usages and emerging types of PUFs that feature appealing properties not usually found in earlier PUFs. In the long term, we envision that the innovations in this area will continue in a direction that will result in commercially viable security-enhanced integrated products with PUF as an integral part of the networked terminals and mobile devices.

References

1. Agarwal, A., Blaauw, D., Zolotov, V.: Statistical timing analysis for intra-die process variations with spatial correlations. In: IEEE/ACM International Conference on Computer-Aided Design, San Jose, pp. 900–907 (2003)
2. Armatix. Armatix ip1 limited edition set. http://www.armatix.us/iP1-Limited-Edition.804.0.html?&L=7 (2014). Accessed 23 April 2014
3. Ayat, M., Atani, R.E., Mirzakuchaki, S.: On design of PUF-based random number generators. Int. J. Netw. Secur. Appl. **3**(3), 30–40 (2011)
4. Bardell, P.H., McAnney, W.H., Savir, J.: Built In Test for VLSI: Pseudorandom Techniques. Wiley, New York (1987)
5. Baumgarten, A., Tyagi, A., Zambreno, J.: Preventing IC piracy using reconfigurable logic barriers. http://www.univ-st-etienne.fr/salware/Bibliography_Salware/ICProtection/article/Baumgarten2010.pdf (2010). Accessed 23 April 2014

6. Beckmann, N., Potkonjak, M.: Hardware-based public-key cryptography with public physically unclonable functions. In: Information Hiding, Darmstadt, pp. 206–220. Springer, New York (2009)
7. Bhargava, M., Mai, K.: A high reliability PUF using hot carrier injection based response reinforcement. In: Workshop on Cryptographic Hardware and Embedded Systems, Santa Barbara, pp. 90–106 (2013)
8. Bhargava, M., Cakir, C., Mai K.: Reliability enhancement of bi-stable PUFs in 65nm bulk CMOS. In: IEEE International Symposium on Hardware-Oriented Security and Trust, San Francisco, pp. 25–30 (2012)
9. Bohm, C., Hofer, M., Pribyl, W.: (2011) A microcontroller SRAM-PUF. In: IEEE International Conference on Network and System Security, Milan, pp.269–273 (2012)
10. Canada, I.: Archived – principles for electronic authentication. A Canadian framework. http://www.ic.gc.ca/eic/site/ecic-ceac.nsf/eng/h_gv00240.html. Accessed 23 April 2014
11. Charmaine, K.: Random number generators: An evaluation and comparison of random.org and some commonly used generators. Technical report, Trinity College Dublin (2005)
12. Chen, Q., Csaba, G., Ju X., Natarajan, S., Lugli, P., Stutzmann, M., Schlichtmann, U., Ruhrmair U.: Analog circuits for physical cryptography. In: IEEE International Symposium on Integrated Circuits, pp. 121–124 (2009)
13. Chen, Q., Csaba, G., Lugli, P., Schlichtmann, U., Stutzmann, M., Ruhrmair, U.: Circuit-based approaches to SIMPL systems. J. Circ. Syst. Comput. **20**(01), 107–123 (2011)
14. Choi, W., Kim, S., Kim, Y., Park, Y., Ahn, K.: PUF-based encryption processor for the RFID systems. In: IEEE International Conference on Computer and Information Technology, Bradford, pp. 2323–2328 (2010)
15. Cortez, M., Hamdioui, S., van der Leest, V., Maes, R., Schrijen, G.J.: Adapting voltage ramp-up time for temperature noise reduction on memory-based PUFs. In: IEEE International Symposium on Hardware-Oriented Security and Trust, Austin, pp. 35–40 (2013)
16. Danger, J.L., Guilley, S., Hoogvorst, P.: High speed true random number generator based on open loop structures in FPGAs. Microelectron. J. **40**(11), 1650–1656 (2009)
17. Dejun, L., Zhen, P.: Research of true random number generator based on PLL at FPGA. In: International Workshop on Information and Electronics Engineering, Harbin, Heilongjiang, vol. 29, pp. 2432–2437 (2012)
18. DigilentInc (2014) Digilent adept for windows. http://www.digilentinc.com/ (2014). Accessed 15 March 2014
19. Eichhorn, I., Koeberl, P., van der Leest, V.: Logically reconfigurable PUFs: Memory-based secure key storage. In: ACM workshop on Scalable Tusted Computing, Chicago, pp. 59–64 (2011)
20. Fedbizoppsgov (2014) Vanishing programmable resources (vapr). https://www.fbo.gov/index?s=opportunity&mode=form&tab=core&id=880ecdf170660730fe0fb8745f5c2bec (2014). Accessed 23 April 2014
21. Fruhashi, K., Shiozaki, M., Fukushima, A.: The arbiter-PUF with high uniqueness utilizing novel arbiter circuit with delay-time measurement. In: IEEE International Symposium on Circuits and Systems, Rio de Janeiro, pp. 2325–2328 (2011)
22. Gainsford, P.: DS2703 SHA-1 Battery Pack Authentication IC. Maxim Integrated, San Jose (2014)
23. Garg, A., Kim, T.T.: Design of SRAM PUF with improved uniformity and reliability utilizing device aging effect. In: IEEE International Symposium on Circuits and Systems, Melbourne, pp 1941–1944 (2014)
24. Gassend, B.: Physical random functions. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2003)
25. Gassend, B., Clarke, D., Van Dijk, M., Devadas, S.: Controlled physical random functions. In: Annual Computer Security Applications Conference, Las Vegas, pp. 149–160 (2002)
26. Gassend, B., Clarke, D., vanDijk, M., Devadas, S.: Silicon physical random functions. In: ACM Conference on Computer and Communications Security, Washington, pp. 148–160 (2002)

27. Guler, U., Ergun, S.: A high speed IC random number generator based on phase noise in ring oscillators. In: IEEE International Symposium on Circuits and Systems, Paris, pp. 425–428 (2010)
28. Hamming R.: Error detecting and error correcting codes. Bell. Syst. Tech. Syst. **29**, 147–160 (1950)
29. Holcomb, D.E., Burleson, P.W., Fu, K.: Power-up SRAM state as an identifying fingerprint and source of true random numbers. IEEE Trans. Comput. **58**(9), 1198–1210 (2009)
30. Holotyak, T., Voloshynovskiy, S., Koval, O., Beekhof, F.: Fast physical object identification based on unclonable features and soft fingerprinting. In: IEEE International Conference on Acoustics, Speech and Signal Processing, Prague, pp. 1713–1716 (2011)
31. Hori, Y., Yoshida, T., Katashita, T., Satoh, A.: Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs. In: IEEE International Conference on Reconfigurable Computing and FPGAs, Cancun, pp. 298–303 (2010)
32. Ivaniuk, A.A.: Application of configurable pulse generator for FPGA identification. Informatika (Informatics) (**4**), 113–123, (2011, in Russian)
33. Ivaniuk, A.A.: Embedded Systems Design. Bestprint, Minsk (2012, in Russian)
34. Ivaniuk, A.A.: Physical unclonable functions unique FPGA chip identification algorithm hardware implementation. In: International Conference Information Technologies and Systems, Minsk, pp. 184–185, (2013, in Russian)
35. Ivaniuk, A.A., Yarmolik, V.N.: Testable Design of Digital Devices. Bestprint, Minsk (2006, in Russian)
36. Kang, H., Hori, Y., Satoh, A.: Performance evaluation of the first commercial PUF-embedded RFID. In: IEEE Global Conference on Consumer Electronics, Tokyo, pp. 5–8 (2012)
37. Katzenbeisser, S., Kocabaş, Ü, van der Leest, V., Sadeghi, A.R., Schrijen, G.J., Wachsmann C.: Recyclable PUFs: Logically reconfigurable PUFs. J. Cryptogr. Eng. **1**(3), 177–186 (2011)
38. Kohlbrenner P., Gaj K.: An embedded true random number generator for FPGAs. In: ACM International Symposium on Field Programmable Gate Arrays, Monterey, pp. 71–78 (2004)
39. Koushanfar, F.: Integrated circuits metering for piracy protection and digital rights management: An overview. In: Great Lakes Symposium on VLSI, Lausanne, pp. 449–454 (2011)
40. Kumar, S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P.: The butterfly PUF: Protecting IP on every FPGA. In: IEEE International Symposium on Hardware-Oriented Security and Trust, Anaheim, pp. 67–70 (2008)
41. Kursawe, K., Sadeghi, A., Schellekens, D., Skoric, B., Tuyls, P.: Reconfigurable physical unclonable functions-enabling technology for tamper-resistant storage. In: IEEE International Symposium on Hardware-Oriented Security and Trust, San Francisco, pp. 22–29 (2009)
42. Kwok, S., Lam, E.: Fpga-based high-speed true random number generator for cryptographic applications. In: TENCON IEEE Region 10 Conference, Hong Kong, pp. 1–4 (2006)
43. Lee, J.W, Lim, D., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for identication and authentication applications. In: IEEE Symposium on VLSI Circuits, Honolulu, pp. 176–179 (2004)
44. Lehtonen, M., Staake, T., Michahelles, F., Fleisch, E.: From identification to authentication – a review of RFID product authentication techniques. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.330.8769&rep=rep1&type=pdf (2008). Accessed 23 April 2014 (2008)
45. Lin, S., Costello, D.: Error Control Coding, Fundamentals and Applications. Pearson Education International, New Jersey (2004)
46. Lofstrom, K.: (2007) ICID a robust, low cost integrated circuit identification method. http://www.siidtech.com/white9.pdf(2014). Accessed 23 April 2014
47. Maes, R., Tuyls, P., Verbauwhede, I.: Intrinsic PUFs from flip-flops on reconfigurable devices. http://www.cosic.esat.kuleuven.be/publications/article-1173.pdf (2008). Accessed 30 Jan 2014
48. Maes, R., Rozic, V., Verbauwhede, I., Koeberl, P., Van der Sluis, E., van der Leest, V.: Experimental evaluation of physically unclonable functions in 65 nm CMOS. In: IEEE European Solid-State Circuit conference, Bordeaux, pp. 486–489 (2012)

49. Maiti, A., Schaumont, P.: Improving the quality of a physical unclonable function using configurable ring oscillators. In: IEEE International Conference on Field Programmable Logic and Applications, Prague, Czech Republic, pp. 703–707 (2009)
50. Maiti, A., Nagesh, R., Reddy, A., Schaumont, P.: Physical unclonable function and true random number generator: a compact and scalable implementation. In: ACM Great Lakes Symposium on VLSI, Boston Area, pp. 425–428 (2009)
51. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Techniques for design and implementation of secure reconfigurable PUFs. ACM Trans on Reconfigurable Technol. Syst. **2**(1), 5 (2009)
52. Mansouri, S.S., Dubrova, E.: Ring oscillator physical unclonable function with multi level supply voltages. In: International IEEE Conference on Computer Design, Montreal, pp. 520–521 (2012)
53. Marsaglia, G.: Diehard: A battery of tests of randomness. http://stat.fsu.edu/_geo (1996). Accessed 28 Jan 2014 (1996)
54. Meguerdichian, S., Potkonjak, M.: Matched public PUF: ultra low energy security platform. In: IEEE/ACM International Symposium on Low Power Electronics and Design, Fukuoka, pp. 45–50 (2011)
55. Moon, T.: Error Correction Codeing: Mathmatical Methods and Algorithms. Wiley, New York (2005)
56. Moreira, J.: Essentials of Error-Control Coding. Wiley, New York (2006)
57. Okumura, S., Yoshimoto, S., Kawaguchi, H., Yoshimoto, M.: A physical unclonable function chip exploiting load transistors' variation in SRAM bitcells. In: IEEE Asia and South Pacific Design Automation Conference, Yokohama, pp. 79–80 (2013)
58. Pappu S.R.: Physical one-way functions. PhD thesis, School of Architecture and Planning, Massachusetts Institute of Technology (2001)
59. Pappu, R., Recht B., Taylor J., Gershenfeld N.: Physical one-way functions. Science **297**, 2026–2030 (2002)
60. Pasupathinathan, V.: Hardware-based identification and authentication systems. PhD thesis, Faculty of science, Macquarie University (2009)
61. Porsch, R.: Protecting devices by active coating. J. Univ. Comput. Sci. **4**(7), 652–668 (1998)
62. Potkonjak, M., Meguerdichian, S., Nahapetian, A., Wei, S.: Differential public physically unclonable functions: architecture and applications. In: ACM Design Automation Conference, San Diego, pp. 242–247 (2011)
63. Qu, G., Yin, C.E.: Temperature-aware cooperative ring oscillator PUF. In: IEEE International Workshop on Hardware-Oriented Security and Trust, San Francisco, pp. 36–42 (2009)
64. Ranasinghe D., Lim D., Devadas S., Abbott D., Cole P.: Random numbers from metastability and thermal noise. Electron. Lett. **41**(16), 13–14 (2005)
65. Randomorg (2014) Introduction to randomness and random numbers. http://www.random.org/randomness/(2013). Accessed 03 Feb 2013
66. Rose G.S., Rajendran J., McDonald N.R., Karri R., Potkonjak M., Wysocki B.T.: Hardware security strategies exploiting nanoelectronic circuits. In: IEEE Asia and South Pacific Design Automation Conference, Yokohama, pp. 368–372 (2013)
67. Roy, J.A., Koushanfar F., Markov I.L. EPIC: Ending piracy of integrated circuits. Computer **43**(10), 30–38 (2007)
68. Rührmair, U.: SIMPL systems: On a public key variant of physical unclonable functions. IACR Cryptology ePrint Archive 2009:255 (2009)
69. Rührmair, U.: SIMPL systems, or: can we design cryptographic hardware without secret key information? In: Conference on Current Trends in Theory and Practice of Computer Science, Novy Smokovec, Slovakia, pp. 26–45 (2011)
70. Rührmair, U.: SIMPL systems as a keyless cryptographic and security primitive. In: Cryptography and Security: From Theory to Applications. Spinger, New York , pp. 329–354 (2012)
71. Rührmair, U., Sölter, J., Sehnke, F.: On the foundations of physical unclonable functions. http://eprint.iacr.org/2009/277.pdf (2009). Accessed 23 April 2014
72. Rührmair, U., Chen, Q., Stutzmann, M., Lugli, P., Schlichtmann, U., Csaba, G.: Towards electrical, integrated implementations of SIMPL systems. In: International Workshop on

Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices, Passau, pp. 277–292 (2010)
73. Rührmair, U., et al.: An attack on PUF-based session key exchange and a hardware-based countermeasure: Erasable PUFs. In: International Conference on Financial Cryptography and Data Security, Divi Flamingo Beach Resort, Bonaire, pp. 190–204 (2012)
74. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S.: A Statistical Test Suite For Random And Pseudorandom Number Generators For Cryptographic Applications. NIST Special Publication 800–22, Gaithersburg (2010)
75. Sadr, A., Zolfaghari-Nejad, M.: Physical unclonable function (puf) based random number generator. Adv. Comput. Int. J. **3**(2), 139–145 (2012)
76. Selimis G., Konijnenburg M., Ashouei M., Huisken J., de Groot H., van der Leest V., Schrijen G.J., van Hulst M., Tuyls P.: Evaluation of 90nm 6t-SRAM as physical unclonable function for secure key generation in wireless sensor nodes. In: IEEE International Symposium on Circuits and Systems, Rio de Janeiro, pp. 567–570 (2011)
77. StatSoftInc. Statistica line of software. http://www.statsoft.com/ (2014). Accessed 15 March 2014
78. Su, Y., Holleman, J., Otis, B.P.: A digital 1.6 pj/bit chip identification circuit using process variations. IEEE J. Solid-State Circuits **43**(1), 69–77 (2008)
79. Suh, G., O'Donnell C., Devadas S.: Aegis: A single-chip secure processor. IEEE Des. Test Comput. **24**(6), 570–580 (2007)
80. Sunar, B., Martin, W., Stinson, D.: A provably secure true random number generator with built-in tolerance to active attacks. IEEE Comput. **56**(1), 109–119 (2007)
81. Tirkel, A.Z., Rankin, G.A., Van-Schyndel, R.M., Ho, W.J., Mee, N.R.A. Osborne, C.F.: Electronic water mark. In: IEEE International Conference on Digital Image Computing: Techniques and Applications, Sydney, pp. 666–673 (1993)
82. Tsoi, K., Leung, K., Leong, P.: High performance physical random number generator. IET Comput. Digit. Tech. **4**(1), 349–352 (2007)
83. Tuyls, P., Skoric, B., Kevenaar, T.: Security with Noisy Data. Springer, New York (2007)
84. Tuzzio, N., Xiao, K., Zhang, X., Tehranipoor, M.: A zero-overhead IC identification technique using clock sweeping and path delay analysis. In: ACM Great Lakes Symposium on VLSI, Salt Lake Cit, pp. 95–98 (2012)
85. Varchola, M., Drutarovsky, M., Fouquet, R., Fischer, V.: Hardware platform for testing performance of TRNGs embedded in actel fusion FPGA. In: IEEE International Conference Radioelektronika, Prague, pp. 1–4 (2008)
86. VerayoInc. Verayo inc. http://www.verayo.com/ (2014). Accessed 15 March 2014
87. Vivekraja, V., Nazhandali, L.: Feedback based supply voltage control for temperature variation tolerant PUFs. In: IEEE International Conference on VLSI Design, Chennai, pp. 214–219 (2011)
88. Wendt, J.B., Potkonjak, M.: The bidirectional polyomino partitioned PPUF as a hardware security primitive. In: IEEE Global Conference on Signal and Information Processing, Austin, pp. 257–260 (2013)
89. XilinxInc. Xilinx ise design tools. http://www.xilinx.com/ (2014). Accessed 15 March 2014
90. Xin, X., Kaps, J., Gaj, K.: A configurable ring-oscillator-based PUF for xilinx FPGAs. In: IEEE Euromicro Conference on Digital System Design, Architectures, Methods and Tools, Oulu, pp. 651–657 (2011)
91. Yarmolik, V., Demidenko, S.: Pseudorandom Signals Generation and Use in the Control and Test Systems. Nauka and Technika, Minsk (1986, in Russia)
92. Yarmolik, V.N., Vashinko, Y.G.: Physical unclonable functions. Informatika (Informatics) **2**, 92–103 (2011, in Russian)
93. Yu, M.D., Devadas, S.: Correction for physical unclonable functions. IEEE Des. Test Comput. **27** 48–65 (2010)
94. Yu, H., Leong, P.H.W., Xu, Q.: An FPGA chip identification generator using configurable ring oscillators. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **20**(12), 2198–2207 (2012)

95. Zalivaka, S.S., Ivaniuk, A.A.: Combined physical unclonable function circuit implementation for generation true random number sequences. Doklady BGUIR **7**(77), 37–43 (2013, in Russian)
96. Zalivaka, S.S., Ivaniuk, A.A.: Physical unclonable functions as entropy source to build true random number generator. In: Belarus-Korea forum Science. Innovation. Production, Minsk, pp. 87–88 (2013)
97. Zalivaka, S.S., Ivaniuk, A.A.: True random number sequences generation with arbiter physical unclonable function. In: International Conference Information Technologies and Systems, Minsk, pp. 204–205, (2013, in Russian)
98. Zalivaka S.S., Ivaniuk A.A. The use of physical unclonable functions for true random number sequences generation. Autom. Control Comput. Sci. **47**(3), 156–164 (2013)
99. Zhang, X.: On-chip structures and techniques to improve the security, trustworthiness and reliability of integrated circuits. http://digitalcommons.uconn.edu/cgi/viewcontent.cgi?article=6219&context=dissertations. Accessed 23 April 2014 (2013)
100. Zhang, L., Kong, Z.H., Chang, C.H.: PCKGen: a phase change memory based cryptographic key generator. In: IEEE International Symposium on Circuits and Systems, Beijing, pp. 1444–1447 (2013)
101. Zhang, L., Fong, X., Chang, C.H., Kong, Z.H., Roy, K.: Highly reliable memory-based physical unclonable function using spin-transfer torque MRAM. In: IEEE International Symposium on Circuits and Systems, Melbourne, VIC (2014, to appear)
102. Zhang, L., Kong, Z.H., Chang, C.H., Cabrini, A., Torelli, G.: Exploiting process variations and programming sensitivity of phase change memory for reconfigurable physical unclonable functions. IEEE Trans. Inf. Forensics Secur. **9**, 921–932 (2014)

Digital Bimodal Functions and Digital Physical Unclonable Functions: Architecture and Applications

Teng Xu and Miodrag Potkonjak

Abstract Security and low power have emerged to become two essential requirements to modern design. The rapid growth of small form, mobile, and remote sensor network systems require secure and ultra-low power data collection and communication solutions due to their energy constraints. The physical unclonable functions (PUFs) have emerged as a popular new type of modern security primitive. They have the properties of low power/energy, small area, and high speed. Moreover, they have excellent security properties and are resilient against physical and side-channel attacks. However, traditional PUFs have two major problems. The first is that the current designs are analog in nature and lack stability in environmental and operational variations, e.g., supply voltage and temperature. The second is that due to the analog nature, the analog PUFs are difficult to be integrated into existing digital circuitry.

In order to leverage the disadvantages of traditional analog PUF, we have proposed two new security primitives, respectively the digital bimodal function (DBF) and the digital PUF. The proposed security primitives preserve all the good properties of traditional analogy PUFs and are stable in the same sense that digital logic is stable. Moreover, both design can be easily integrated into existing digital circuitry. The key idea of DBF is to build a mapping of randomly generated Boolean functions that has two forms: $f_{compact}$ and $f_{complex}$, among which $f_{compact}$ can be computed rapidly and requires only a small amount of energy while $f_{complex}$ can only be computed using a very high amount energy, hardware resources, and an unacceptable amount of time. The performance difference can be applied to enable security protocols. The digital PUF is one more step beyond the DBF, it is designed on the top of DBF to make the DBF design to be unclonable. The key observation is that for any analog delay PUF, there is a subset of challenge inputs for which the PUF output is stable regardless of operational and environmental conditions. We use only such stable inputs to initialize the look-up tables (LUTs) in DBFs that are configured in such a way that the overall structure is unclonable.

We summarize the goal of designing DBF and digital PUF by identifying the architectural, security, and application desiderata. The architectural desiderata

T. Xu • M. Potkonjak (✉)
University of California, Los Angeles, Los Angeles, CA, USA
e-mail: xuteng@cs.ucla.edu; miodrag.potkonjak@gmail.com

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_3

include (1) low energy, delay, and area costs; (2) stability against temperature and voltage variations. The security desiderata include (3) resiliency against security attacks; (4) high outputs randomness; (5) low inputs-outputs correlations. Finally, the application desiderata include (6) small computation, low bandwidth secure protocols.

1 Introduction

The rapid proliferation of mobile systems and devices that operate in potentially hostile environments has elevated security to be one of the most important design metrics. For example, security is essential in smart phones, laptops, and wireless sensor networks. Classical software-based public-key cryptography provides a spectrum of elegant and powerful security protocols. However, it is also subject to several important limitations and drawbacks. The most important drawbacks include susceptibility to physical and side channel attacks and high implementation and energy costs.

The physical unclonable function (PUF) is a cryptographic primitive that has been suggested for many secure domains due to its low power requirements. PUFs are physical devices that have a random but deterministic mapping of inputs to outputs. Their unclonability and functionality are often inextricably tied to the physical characteristics of the device components (e.g. gate delay, leakage energy). While PUFs receive and generate digital inputs and outputs, they are analog in nature due to their reliance and design based on their inherent physical characteristics. Thus, current PUFs have many limitations. The most limiting of which includes susceptibility to environmental and operational conditions. Many PUFs, including the standard delay-based PUF require arbiters to operate. These memory components limit the PUF in terms of placement and coordination in circuitry since their outputs cannot be used directly in the current clock cycle like a combinational module, but instead, require an additional clock cycle to be used.

These main limitations can be removed by creating a security primitive purely in digital domain. The digital security primitive must be stable in the same sense that digital logic is stable against environmental and operational conditions and must produce deterministic outputs for all input vectors. The digital security primitive must be integrable with existing combinational logic without requiring additional clock cycles to use its outputs. And lastly, the digital security primitive must be flexible in the sense that its structure can be altered for different tradeoffs between security, energy, and delay as required by the pertinent task.

In this chapter, we present two security primitives with such characteristics, respectively DBF and digital PUF. The essential idea behind the DBF is to have two forms of a function in which one form is fast and compact ($f_{compact}$) and the other is slow and complex ($f_{complex}$). Both forms have exactly the same functionality, (i.e. given the same input, both forms produce the same output). Figure 1 demonstrates an example of the FPGA-based implementation of $f_{compact}$. The architecture is

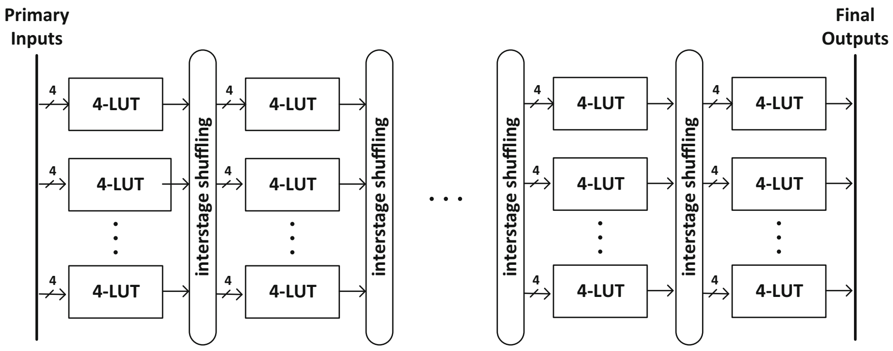


Fig. 1 Example of an FPGA-based DBF LUT network

composed of a randomly connected FPGA network. The hierarchical structure is constructed by feeding the output of the previous stage's LUTs as the input to the next stage. Meanwhile, we use the direct mapping between the primary input and the final output as $f_{complex}$ of the DBF. As a result, both forms have exactly the same functionality. However, as the number of primary inputs and stages in the LUT network grows, the hardware implementation of $f_{compact}$ stays in a relatively compact form while $f_{complex}$ grows exponentially.

The second security primitive digital PUF incorporates the standard analog delay PUF into the design of DBF. The key observation is that for any analog delay PUF, there is a subset of challenge inputs for which the PUF output is stable regardless of operation and environmental conditions. We use only such stable inputs to initialize the look-up tables (LUTs) in the DBF so that the digital PUF is formed. We claim that our design forms a digital PUF in the following sense. In the first place, the basic idea of our proposed design is to utilize the digital functionality of the pseudorandomly connected LUT network to generate multi-input and multi-output mappings. However, traditional PUFs rely on the effect of process variation in the analog properties of the circuit (e.g., delay, frequency, leakage) to generate unique outputs. In our design, although we also utilize the delay-based PUF to ensure unclonability, the responses are only used to initialize the LUT cells of the DBF. Another important argument is that our FPGA-based digital PUF is not subject to operational and environmental conditions. For instance, it is a known fact that many analog systems are highly sensitive to variations, especially temperature, and supply voltages. Our FPGA-based digital PUF completely removes this problem because of the digital nature of the DBF and the utilization of the stable challenge-response pairs for the part of the delay-based PUF.

Our technical goal is to propose DBF and digital PUF as two new types of low-power hardware security primitives. Since digital PUF inherits the properties of DBF and shares the similar architecture, besides, it has the advantage of unclonablility, we briefly compare digital PUF with traditional cryptographic cyphers, analog PUFs. Compared to the traditional cryptographic cyphers, digital PUF has the advantage of low-power, low-area, and high-speed. E.g, using our proposed structure of LUT networks, the encryption/decryption only requires one clock cycle computation. Compared to traditional analog PUFs, digital PUF resolves the problem of instability by completely operating in the digital domain. To be more specific, it utilizes the digital logic functions to build the inputs-outputs mapping. Consequently, it is resilient to the environmental and operational variations.

In this chapter, we first review the previous work on PUFs, then we demonstrate the FPGA-based architecture of the DBF and the digital PUF. Afterwards, we analyze the security of the DBF and digital PUF by applying the NIST randomness benchmark test suite [1] and demonstrating that it passes all the tests. We also analyze the outputs of both security primitives using the security principles of confusion and diffusion, as presented by Shannon [2], through demonstration of the avalanche criterion. Lastly, we explore two important security protocols. Our first protocol is public key communication with DBF. It utilizes the unique property of DBF to enable low-energy, high-speed, small-area public key communication. The second protocol is remote trust, which utilizes digital PUF to enable authentication between parties remotely.

2 Related Work

In this section, we first review the previous works on PUFs, then we give more specific introductions on various types of public PUFs (PPUFs).

2.1 Physical Unclonable Functions

Pappu et al. introduced the concept of the first PUF and demonstrated it using mesoscopic optical systems [3]. Devadas' research group at MIT developed the first family of silicon PUFs through the use of intrinsic process variation in deep submicron integrated circuits [4]. Guajardo and his coworkers at Philips Research in Eindhoven demonstrated how PUFs can create unique startup values in SRAM cells [5]. Consequently a great variety of technologies were used for PUF creation including IC interconnect networks, thyristors, memristors, and several nanotechnologies. Although a variety of PUF structures have been proposed, arbiter-based (APUF) [4], ring oscillator-based (RO-PUF) [6], and SRAM PUFs [5] are by far most popular.

PUFs were immediately applied to a number of applications including authentication, cryptographic key generation and secure storage [7], anti counterfeiting [8], FPGA intellectual property (IP) protection [9], remote enabling and disabling of integrated circuits [10], remote trusted sensing [11, 12], and random number generator [13]. PUFs are also used in conjunction with traditional creation and operation of remote secure processors [14]. The security role of the PUF has been greatly enhanced with several proposals for employing PUFs in public key security protocols in systems such as the public PUF (PPUF), SIMPL, and one time pads [15, 16]. Recently, the matched PPUF (mPPUF) [17] has been proposed as a new public key security primitive. mPPUF uses both process variation and device aging to create pairs of identical PPUFs that can be matched only with negligible small probability. It is a very energy efficient security primitive that can be used in a variety of cryptographic protocols.

Our DBF/digital PUF is also a type of PPUF which enables public key cryptography. In the next part of related work, we will give more specific introduction on various types of PPUFs and compare them with our DBF/digital PUF.

2.2 *Public PUFs*

PPUFs have extended the practicality of PUFs by enabling the creation of public key protocols. While PUFs require that their characterization and structure remain hidden and secret, the PPUF design and characterization is disclosed to the public. In this way, the design itself becomes the public key. PPUFs have small area footprint and orders of magnitude lower energy consumption than their traditional cryptographic counterparts. The applications of PPUFs in securing internet of things are proposed in [18].

2.2.1 XOR Network Delay PUF

Beckmann et al. proposed the first PPUF model along with accompanying protocols for public key cryptography [15]. The public key consisted of the complete characterization of the design, including gate-level characteristics, such as leakage energy and delay. Due to the effects of process variation, inherent doping concentrations variances and line-edge roughnesses manifested as different values of effective channel lengths and threshold voltages which ultimately effect leakage energy and delay of each transistor. In this way, the public key was random, and unclonable, however, still able to be simulated, although very arduous to do so due to the design.

The architecture of Beckmann's PPUF is a gridded network of XOR gates. Due to inherent intrinsic manufacturing variability, the physical characteristics of each XOR gate differ. Specifically, due to variations in doping concentrations and line edge roughness, differences in threshold voltages and effective channel lengths emerge. When sending an input through the gates, the rising edges will race throughout the gridded network. Each XOR gate will transition upon the arrival of a new rising or lowering edge and emit the output corresponding to its input at that particular time. These signals will propagate throughout the circuit, causing multiple transitions at each XOR gate. The input challenge is a combination of both the input vectors ($x(0)$, $x(1)$) as well as a time delay (t) at which to read the outputs of the network.

The design of this PPUF takes advantage of the glitching effects of multiple propagating and delayed signals throughout the XOR network. This architecture also requires ultra accurate, ultra precise, and ultra high frequency clocks in order to operate on the physical PPUF, and, for larger PPUFs, requires much longer simulation times for the communicating parties wishing to initiate authorized contact with a PPUF owner.

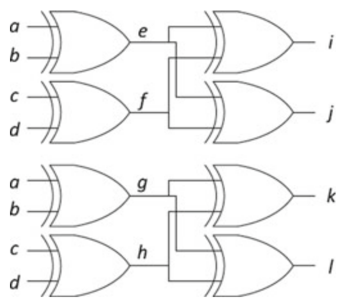
2.2.2 Differential PUF

The differential PPUF eliminates the need for ultra accurate clock manipulation for high precision timing as well as long simulation times [19]. Like its predecessor, the unclonability of the differential PPUF relies on the inherent randomness in manufacturing variability, specifically manifesting as variances in gate delays. A key novelty of this architecture is that the challenge vector is reduced from two input vectors plus a timestamp to a single input vector. This eliminates the need for accurate clock capturing of glitch temporal characteristics because it only requires the measurement of the frontier signal.

Consider the differential PPUF booster example depicted in Fig. 2. If the input switches from 0000 to 0101, output *i* will switch at times 6 and 11, *j* at times 9 and 12, *k* at times 8 and 9, and *l* at times 7 and 12. By placing an arbiter with inputs from *i* and *k* and a second arbiter with inputs from *j* and *k*, we eliminate the need for high precision timing by only capturing the first winner of the two paths. Hence, only frontier signals are necessary. However, since one has to simulate only these frontier signals, an architecture in which one can predict which frontier signals will not cause transitions is not secure. Thus, in addition to booster cells, the differential PPUF includes represser cells consisting of a NAND gate network to terminate subsets of propagating signals in an unpredictable manner, such as the one depicted in Fig. 3. Together, the alternation of booster cells followed by represser cells creates a highly non-linear system that is exponentially hard to simulate with a linear size increase.

Fig. 2 Differential PPUF booster cell example. The delay of a rising edge from input *i* to output *j* is denoted by δ_{ij}

δ_{ae}	3	δ_{ei}	7
δ_{be}	4	δ_{fi}	2
δ_{cf}	5	δ_{ej}	5
δ_{df}	4	δ_{fj}	8
δ_{ag}	4	δ_{gk}	5
δ_{bg}	5	δ_{hk}	4
δ_{ch}	8	δ_{gl}	7
δ_{dh}	4	δ_{hl}	3



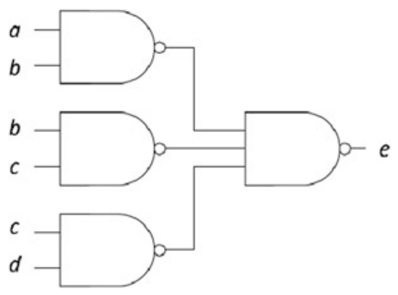


Fig. 3 Differential PPUF represser cell example

2.2.3 Device Aging and Matched PUFs

All previously proposed PPUFs, including the differential PPUF, are potentially subject to long-term reverse engineering attacks. The device aging-based PPUF design eliminates the possibility of these attacks through dynamic reconfiguration. The key idea is to leverage device aging to alter the PUF's physical properties, thus changing its behavior. Specifically, device aging through techniques such as NBTI can permanently alter the threshold voltages of gates, thus increasing their delay [20].

The key limitation to the original device aging-based PPUF, along with all other previously designed PPUFs, is that they employ a large time gap between execution and simulation to enable public key communication. While each PPUF design provided faster simulation time on the part of the authentication party than its predecessor, the fact remains that at least one participating party requires significant resources for communication in comparison to the participant in possession of the physical PPUF.

The matched PPUF architecture attempts to remove the need for simulation entirely by supplying both communicating parties with physical PPUFs that are globally unique post fabrication, but can be made identical through a novel matching procedure. This procedure is executed in such a way that only the two participating PPUFs become identical while it is probabilistically negligible that a third snooping adversary is able to match as well.

The architecture of the matched PPUF utilizes booster cells and represser cells, similar to those designed for the differential PPUF and depicted in Figs. 2 and 3. The first matched PPUF architecture consisted of h stages of b booster cells followed by r represser cells, and interstage networks connecting them as depicted in Fig. 4. Matching is done post fabrication when two communicating parties, each with their own PPUF, enable, disable, and age their individual sets of gates until a portion of gates are matched between the two of them and their PPUFs now implement the same functionality.

An adversary snooping on the matching protocol is still only able to match 58.3 % of the configuration [17]. Attempting to match the remaining gates through simulation or special purpose hardware is not quick enough to successfully imitate the physical PUF. Furthermore, the task is made even more difficult by increasing the size of the PPUF, thereby increasing the total number of unmatched adversarial gates which has an exponential increase in simulation complexity.

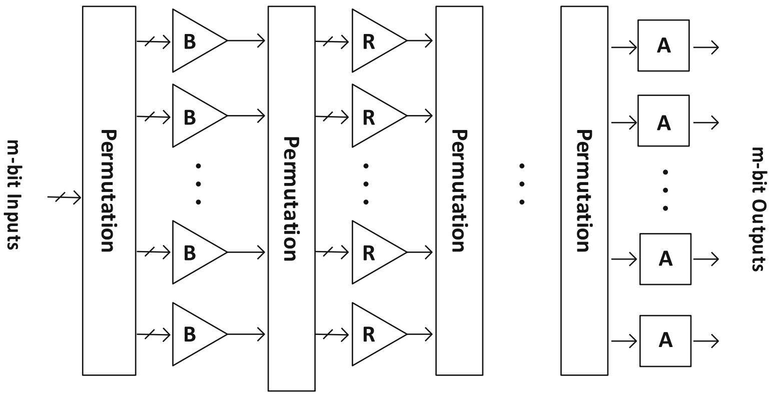


Fig. 4 Device aging-based matched PPUF architecture

Compared to all the previous work on PPUFs, our DBF/digital PUF have the advantage of completely operating in the digital domain. Besides, they have even faster speed and lower energy. Starting from the next section, we formally introduce the DBF/digital PUF. Since digital PUF is built based on the structure of DBF, we first introduce the design of DBF, then explain how we build digital PUF from DBF.

3 Digital Bimodal Functions

We demonstrate the design of DBF in this section. We start from a motivational example, then explain the architecture of DBF for both $f_{compact}$ and $f_{complex}$. Lastly, we compare the two forms of DBF using experimental data.

3.1 A Motivational Example

The concept of the digital bimodal function (DBF) was first proposed by Xu et al. [21]. The essential idea behind the DBF is to represent a set of binary functions in two forms, one which is fast and compact ($f_{compact}$) and the other which is slow and complex ($f_{complex}$). Both forms have exactly the same functionality, in other words, given the same inputs, both forms produce the same outputs.

Equations (1)–(3) illustrate an example of a DBF. As a prerequisite, a_i, b_i, and c_i are binary values, and the function sets f and g are Boolean functions in the form of sums of products (SOP) and/or products of sums (POS) representing $f_{compact}$ and $f_{complex}$, respectively. Equation (1) represents the relationship between a_i and b_i and Eq. (2) represents the relationship between b_i and c_i. Note that each function f has 4 binary inputs assigned in a random and permanent order.

Equation (3) is generated by substituting (1) into (2), yielding a direct relationship between a_i and c_i. Note that substitutions are expanded and simplified so that each sub function in g is in the form of a SOP or a POS. The key observation here is that while both f and g implement the same functionality, f can be computed much more rapidly than g since it is in a compact format in which each subfunction requires only four inputs, while g is in an expanded format in which each subfunction requires up to n variables. It has been shown that the size difference between $f_{compact}$ and $f_{complex}$ increases exponentially with an increase in input variables and additional levels of substitution [21].

In order to visualize the size difference between $f_{compact}$ and $f_{complex}$, we first set up a few premise, then quantify their size difference.

The first premise is that we put both $f_{compact}$ and $f_{complex}$ into the form of a sum of products and simplify them by using the tool Simple Solver. We then compare the total number of products. Note that the simplification procedure reduces the size of the Boolean functions, but only by a constant factor. In Example I, after simplification, $f_{compact}$ can be expressed by the group of functions in (1c), which has 19 products, and $f_{complex}$ can be expressed by the functions in (1e), which have 67 products.

$$\begin{array}{ll} \text{Inputs:} & a_i \in \{0,1\}, i \in \{0,1,2\dots n-1\} \\ \text{Outputs:} & c_i \in \{0,1\}, i \in \{0,1,2\dots n-1\} \\ \text{Variables:} & b_i \in \{0,1\}, i \in \{0,1,2\dots n-1\} \\ & r_j \in \{0,1,2\dots n-1\}, j \in \{0,1,2\dots 8n-1\} \end{array}$$

$$\begin{cases} b_0 = f_0(a_{r_0}, a_{r_1}, a_{r_2}, a_{r_3}) \\ b_1 = f_1(a_{r_4}, a_{r_5}, a_{r_6}, a_{r_7}) \\ b_2 = f_2(a_{r_8}, a_{r_9}, a_{r_{10}}, a_{r_{11}}) \\ \dots \\ b_{n-1} = f_{n-1}(a_{r_{4n-4}}, a_{r_{4n-3}}, a_{r_{4n-2}}, a_{r_{4n-1}}) \end{cases} \tag{1}$$

$$\begin{cases} c_0 = f_n(b_{r_{4n}}, b_{r_{4n+1}}, b_{r_{4n+2}}, b_{r_{4n+3}}) \\ c_1 = f_{n+1}(b_{r_{4n+4}}, b_{r_{4n+5}}, b_{r_{4n+6}}, b_{r_{4n+7}}) \\ c_2 = f_{n+2}(b_{r_{4n+8}}, b_{r_{4n+9}}, b_{r_{4n+10}}, b_{r_{4n+11}}) \\ \dots \\ c_{n-1} = f_{2n-1}(a_{r_{8n-4}}, a_{r_{8n-3}}, a_{r_{8n-2}}, a_{r_{8n-1}}) \end{cases} \tag{2}$$

$$\begin{cases} c_0 = g_0(a_0, a_1, a_3, \dots, a_{n-1}) \\ c_1 = g_1(a_0, a_1, a_3, \dots, a_{n-1}) \\ c_2 = g_2(a_0, a_1, a_3, \dots, a_{n-1}) \\ \dots \\ c_{n-1} = g_{n-1}(a_0, a_1, a_3, \dots, a_{n-1}) \end{cases} \tag{3}$$

Table 1 Size comparison between $f_{compact}$ and $f_{complex}$ with different number of iterations and different number of primary inputs

# of iterations	# of inputs	Avg. # of products ($f_{compact}$)	Avg. # of products ($f_{complex}$)
4	8	23.2±2.8	259.2±16.1
5	10	28.7±3.1	765.6±68.7
6	12	34.6±3.5	3816.0±245.2
7	14	39.0±3.8	11454.8±758.1
8	16	46.7±4.1	49206.4±2684.8
9	18	52.1±4.3	142491.6±7520.1
10	20	57.8±4.6	369656.8±19265.5

The average number of products are tested with 95 % interval confidence

The second premise is that each single sub-function in $f_{compact}$ has at most 4 inputs, just as shown in Example I, which is used to limit the size of $f_{compact}$.

The third one is that the number of outputs is the same as the number of inputs in each iteration, e.g., in Example I, the number of a_i equals to the number of b_i and $c_i, i \in \{0, 1, 2, \ldots, 7\}$.

The last premise is that the number of iterations is half of the number of inputs. While iterations can create size difference between $f_{compact}$ and $f_{complex}$, the size is also limited by the number of inputs. Therefore the number of iterations should be proportional to the number of inputs, as an example, we set it to be half.

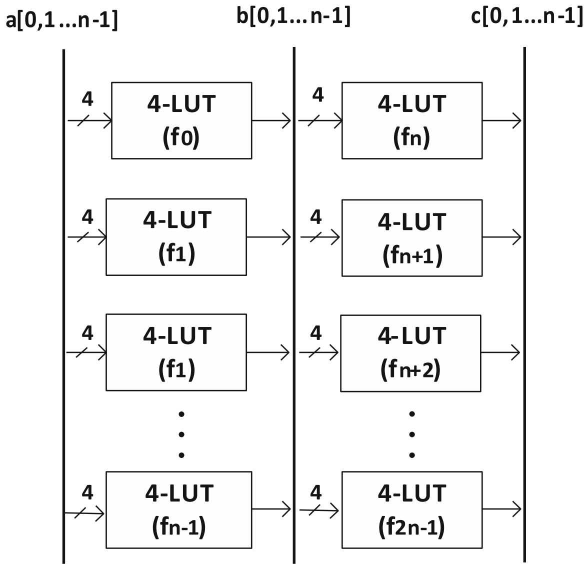
Based on the above definition, with the increase in the number of inputs, we randomly generate a group of functions as $f_{compact}$ and the corresponding $f_{complex}$, then compare the number of products.

The size difference between $f_{compact}$ and $f_{complex}$ determines their difference in computation complexity. According to Table 1, the number of products in $f_{compact}$ grows linearly while the number of products in $f_{complex}$ grows exponentially. For example, when the number of inputs is 20, the average # of products in $f_{compact}$ is 57.8 while the average # of products in $f_{complex}$ reaches approximately 370,000 which is 6,400 times larger. Therefore, by increasing the number of inputs as well as the number of iterations, it is very easy to create a huge computation gap between $f_{compact}$ and $f_{complex}$.

3.2 FPGA-Based Implementation

Figure 5 depicts the FPGA-based implementation of the $f_{compact}$ of the DBF defined in Eqs. (1)–(3). The architecture is composed of two levels of 4-input LUTs. Note that each 4-input LUT implements a 4-input Boolean function from f. A hierarchy structure is constructed by feeding the outputs of the previous level of LUTs to the inputs of the next level of LUTs which is equivalent to the function substitution.

Fig. 5 An example of the FPGA-based DBF $f_{compact}$ LUT network



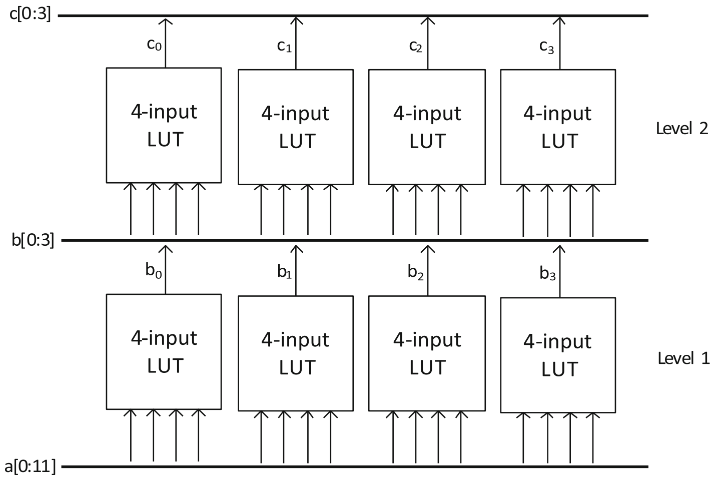


Fig. 6 A combinational logic implementation of DBF $f_{compact}$ LUT network

Therefore, the LUT network directly implements $f_{compact}$ in the DBF. As the number of inputs and the number of levels in the LUT network grow, the expanded form of $f_{complex}$ becomes very difficult to be implemented in hardware (grows exponentially) while $f_{compact}$ remains in a relatively compact form (grows linearly).

An even more compact FPGA implementation is to use sequential logic. The key idea is to iteratively use one level of LUT networks. This requires each level of sub functions in $f_{compact}$ to have the same format. For example, the combinational logic in Fig. 6 can be replaced with the sequential logic in Fig. 7. Compared to

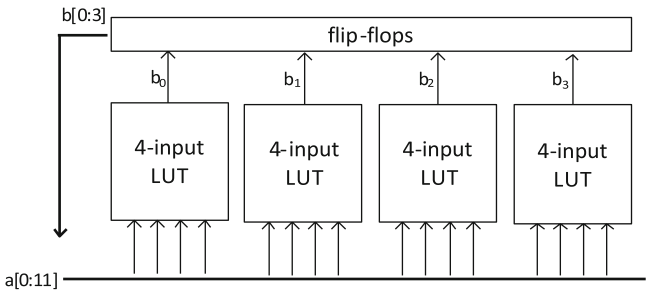


Fig. 7 A sequential logic implementation of DBF $f_{compact}$ LUT network

Table 2 Average synthesis resources compared between DBF form $f_{compact}$ and form $f_{complex}$

Levels	Inputs	$f_{compact}$ LUT #	$f_{complex}$ LUT #
4	8	8	81
5	10	10	396
6	12	12	1616
7	14	14	4353
8	16	16	13576
9	18	18	31155
10	20	20	98282

The Input # does not have to be twice as the circle #, we set it here as an example. The tests are based on the Spartan-3 XC3S50-5 FPGA and synthesized using the Xilinx ISE

the combinational logic, the sequential logic saves the number of LUTs required. It takes iterations (each iteration is corresponding to a clock cycle) to produce the outputs while the number of clock cycles is equal to the levels of LUT in the combinational logic.

Now we consider the hardware implementation of $f_{complex}$. We use the Xilinx ISE Design Suite to synthesize $f_{complex}$ and compare the resources it requires with $f_{compact}$. For a $f_{compact}$ structure with a given number of inputs and cycles, we generate the corresponding $f_{complex}$, then convert it to a netlist and synthesize it to acquire the number of LUTs required in order for it to be implemented on the FPGA. We change the input # in the experiment and set the cycle # to the half of the inputs #. Table 2 indicates that $f_{complex}$ requires many more LUTs than $f_{compact}$ and the difference keeps growing with the increase of the input #, which could easily reach the extent that the hardware implementation of $f_{complex}$ costs so much that it can only be simulated. Through this technique, due to the time difference between implementation and simulation, the time difference between $f_{compact}$ (private key) and $f_{complex}$ (public key) can be further expanded.

Table 3 Comparisons between $f_{compact}$ and $f_{complex}$

	$f_{compact}$	$f_{complex}$
Operation	Implementation	Simulation
Time (8 inputs, 4 iterations)	29.2 ± 3.67(ns)	$(1.18 \pm 0.08) * 10^4$(ns)
Time (10 inputs, 5 iterations)	37.0 ± 4.02(ns)	$(4.33 \pm 0.49) * 10^4$(ns)
Time (12 inputs, 6 iterations)	45.1 ± 5.48(ns)	$(1.63 \pm 0.12) * 10^5$(ns)
Time (14 inputs, 7 iterations)	53.9 ± 5.51(ns)	$(5.77 \pm 0.48) * 10^5$(ns)
Time (16 inputs, 8 iterations)	61.4 ± 6.49(ns)	$(2.31 \pm 0.25) * 10^6$(ns)
Time (18 inputs, 9 iterations)	69.5 ± 7.36(ns)	$(6.75 \pm 0.49) * 10^6$(ns)
Time (20 inputs, 10 iterations)	77.2 ± 7.99(ns)	$(1.61 \pm 0.19) * 10^7$(ns)

The results show an average implementation/simulation time with the standard deviation

3.3 $f_{compact}$ and $f_{complex}$ Comparisons

Based on the above discussion, Table 3 shows the comparisons between $f_{compact}$ and $f_{complex}$. Both of them can be simulated, but only $f_{compact}$ can be implemented by the sequential logic on an FPGA using a reasonable amount of resources. As implementation is generally faster than simulation, we choose to implement $f_{compact}$ rather than to simulate it. When applying 20 inputs and 10 iterations, the implementation of $f_{compact}$ takes only 77.2 ns while the simulation of $f_{complex}$ takes more than 1.6×10^7 ns. The ratio of time difference reaches 2×10^5. Note that for an $f_{compact}$ structure with a larger size, the time difference can grow easily very large. For different applications, the size of $f_{compact}$ structure can vary.

The LUT network architecture based on configurable logic blocks (CLBs) on FPGA provides a very intuitive and simple way to implement $f_{compact}$, while for $f_{complex}$ the implementation on FPGA takes too many resources, requiring it to be simulated. The time and efficiency difference between $f_{compact}$ and $f_{complex}$ on FPGAs, which would be utilized for security purposes, offers us a strong reason to choose FPGA as an ideal platform.

4 Digital PUF

The concept of digital PUF is first proposed by Xu et al. [22, 23]. It is built on the top of the DBF architecture. The key idea is to incorporate the design of DBF with the traditional delay-based PUF to build the digital PUF. To be more specific, there are two major components that compose the digital PUF: a stable delay-based PUF and a lookup table (LUT) network from DBF. In the following parts of the section, we first explain our technique to create stable challenges of delay-based PUF, then we demonstrate the detail architecture and operations to create the digital PUF.

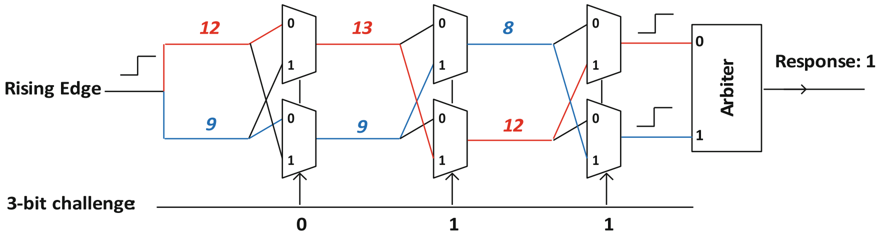


Fig. 8 Applying a 3-bit input challenge to a delay-based PUF. The challenge is intentionally chosen in this example in such a way that the delay difference between the two paths (*red* and *blue*) are maximized

Table 4 Delay differences between all possible paths in the example delay-based PUF in Fig. 8

Challenge	Delay difference
000	3
001	−3
010	−11
011	11
100	−3
101	3
110	−5
111	5

4.1 Stable Challenges and Outputs

Figure 8 depicts an example of a 3-bit delay-based PUF. Each challenge bit controls the inputs of two multiplexers. An output bit is generated by assigning a challenge vector and sending a rising edge through the PUF. The two paths traverse the three delay segments, swapping positions (top and bottom) depending on the input bit at each segment, before arriving at the arbiter which determines the final output. For example, an input challenge of 011 generates the blue and red paths depicted. An arbiter will set its value to 0 or 1 depending on which path (top or bottom) arrives first, effectively selecting the path that has the smaller delay. Table 4 consists of the delay differences between the top and bottom paths for all possible paths in the example PUF in Fig. 8.

A key observation is that for each unique delay-based PUF there exists a set of challenges that produce stable outputs. Consider the situation in which environmental conditions affect the physical characteristics of the circuit. For example, variations in temperature cause variations in individual gate delays, thereby affecting the overall path delays in the analog PUF. Since challenges 011 and 010 result in a large difference in delay between the two racing paths, it is still with high possibility that the red path will have a larger delay compared to the blue path despite the effects temperature may have on the individual gate delays. We label this challenge, and any other challenges that are resilient to such environmental changes, as stable inputs.

For path delay analysis we introduce a *delay ratio* metric, which is defined as the delay differences of two paths divided by the delay of the shorter path. For the purposes of testing, we assume that gate delays follow a normal distribution due to the effects of process variation.

$$Delay\ Ratio = \frac{Delay_{p1} - Delay_{p2}}{min(Delay_{p1}, Delay_{p2})} \tag{4}$$

We use the delay ratio, as defined in Eq. (4), to evaluate the relative delay difference between the two PUF paths. In the following test we assume that the gate delays of the PUF follow a normal distribution due to the effects of process variation.

The distribution of the delay ratio for random challenges on a 32-bit PUF and a 64-bit PUF are depicted in Fig. 9. In both cases, they follow a normal distribution with their means at 0. The standard deviation of the 64-bit PUF is smaller than the 32-bit PUF. To better visualize the probability that the delay ratio is larger than some value, we use Table 5 to show the quantified result.

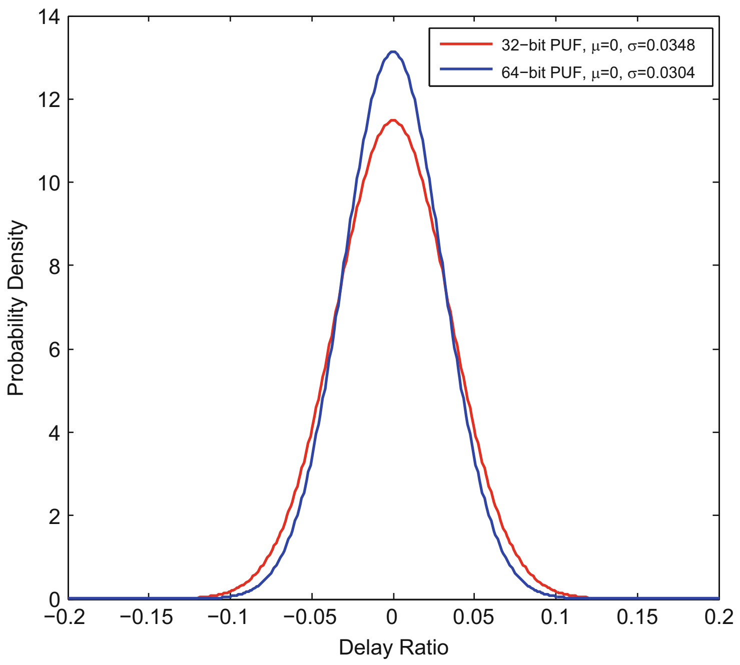


Fig. 9 Distributions of delay ratios for a 32-bit PUF and a 64-bit PUF

Table 5 Probability that the delay ratio (R) is larger than the labelled threshold value for a 32-bit and 64-bit PUF

	$P(R \geq 0.04)$ (%)	$P(R \geq 0.06)$ (%)	$P(R \geq 0.08)$ (%)	$P(R \geq 0.1)$ (%)
32-bit PUF	12.51	4.27	1.07	0.21
64-bit PUF	9.34	2.44	0.43	0.05

Table 6 Probability that outputs of the 32-bit PUF are stable over varying temperatures for different delay ratios

	Delay ratio (T = 300 K)						
Temperature (K)	0.04	0.05	0.06	0.07	0.08	0.09	0.1
250	0.979	0.987	0.994	0.997	1	1	1
350	0.969	0.975	0.989	0.994	0.997	1	1
400	0.937	0.951	0.959	0.977	0.988	0.996	1

Table 7 Probability that outputs of the 64-bit PUF are stable over varying temperatures for different delay ratios

	Delay ratio (T = 300 K)						
Temperature (K)	0.04	0.05	0.06	0.07	0.08	0.09	0.1
250	0.984	0.986	0.996	0.998	1	1	1
350	0.982	0.986	0.993	0.998	1	1	1
400	0.954	0.974	0.986	0.991	0.997	1	1

We simulate and test the stability of our PUF under different environmental temperatures. In our test, we assume that the original distribution of the gate delay D_{old} at temperature 300 K follows the Gaussian distribution $D_{old} \sim \mathcal{N}(\mu, \sigma^2)$. When temperature changes, the delay changes: $D_{change} \sim \mathcal{N}(\alpha\mu, |\alpha|\sigma^2)$ where α is a ratio which is decided by the new temperature, thus yielding the new delay $D_{new} = D_{old} + D_{change}$. We use the Hotspot tool [24] to compute α under different temperatures. For example, α under 400 K is approximately 1. Based on this assumption, under different original delay ratios and after applying temperature changes, we measure the probability that the same challenge produces the same stable outputs. Table 6 shows the results of a 32-bit delay-based PUF. As expected, a higher original delay ratio yields higher probabilities for stable outputs. For example, for an original delay ratio of 0.1, the probability that the PUF output remains stable across temperatures ranging from 250 to 400 K remains 1.

The results of our 64-bit PUF tests are shown in Table 7. Compared to the 32-bit test case, the 64-bit test case demonstrates a similar trend and exhibits even better stability under the same conditions. As long as the original delay ratio reaches a particular threshold (e.g. 0.1 in this experiment), the outputs remain stable for a wide range of temperatures. Hence, we select those challenges that satisfy this delay ratio threshold as the stable challenges.

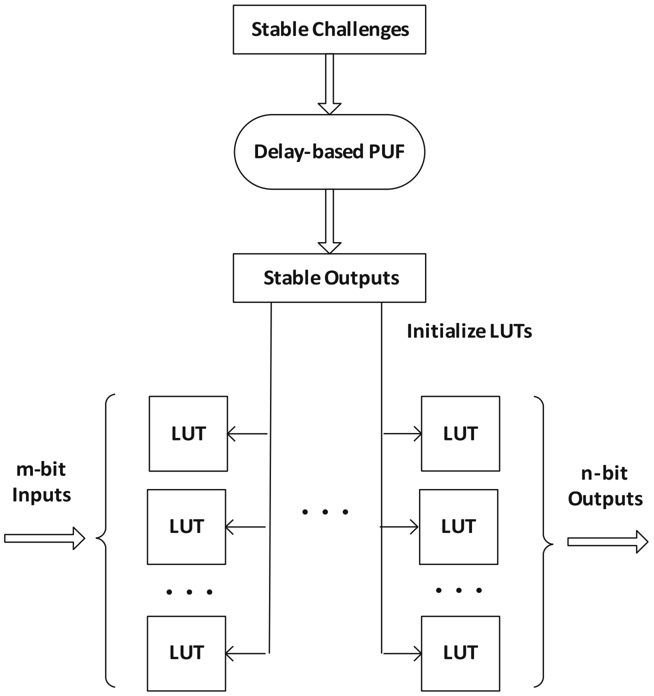


Fig. 10 Architecture of the digital PUF. Note that the stable outputs from the analog PUF are used only once at startup to initialize and configure the LUTs in the DBF

4.2 *Architecture*

Figure 10 depicts the architecture of the digital PUF. At startup, the user selects and applies stable challenge vectors, supplied by the digital PUF manufacturer, to an array of delay-based PUFs. The resultant stable outputs are then used to initialize and configure individual LUT cells in the DBF. This procedure is applied to a random subset of LUT cells, while the remaining cells are initialized by the user. This bifurcation in initialization enables self trust by preventing malicious manufacturers from completely controlling the DBF configuration process.

After PUF initialization, the user generates an input-output mapping for the DBF which serves as a specification of $f_{complex}$. This is easily done by traversing all the possible inputs and generating the corresponding output. The mapping is stored as Boolean functions in both SOP and POS forms.

By applying only stable challenges to the delay-based PUF at initialization we ensure that the entire digital PUF system is completely stable. Furthermore, the intrinsic unclonability of the delay-based PUF along with its integration with the DBF guarantees that the overall architecture is unclonable. Since the delay-based PUF is used only at initialization and is subsequently disregarded and the rest of the digital PUF operation is delegated to the DBF, we inherit the small power, area, and low delay properties of the DBF.

4.3 Operations

In order to use the FPGA-based digital PUF, a set of operations need to be done, we divide them into the following two steps: (1) FPGA configuration and (2) DBF generation.

4.3.1 FPGA Configuration

The essential step before using the DBF is to reconfigure the FPGA for DBF initialization. As we mentioned in the previous section, in every clock cycle the user needs to choose a stable challenge vector $\{C_0, \ldots, C_{k-1}\}$ and feed it to the delay-based PUF. Each challenge is randomly chosen from a pool of stable challenges that is provided by the manufacturer. Note that if we reverse all the challenge bits to $\{\overline{C_0}, \ldots, \overline{C_{k-1}}\}$, the output is reversed too. Hence, there will not be any bias between the number of stable challenges that produce the 0 output and the 1 output. As a result, the output randomness of the delay-based PUF is not reduced because of the use of only stable challenges. Each of the generated output will be used to initialize one cell in a LUT. This procedure is repeated until a random portion of the LUT cells in the DBF are initialized, and the rest are initialized by the user. The reason to choose only a random part of LUTs to initialize is to enable self trust in order to prevent malicious manufacturers, because the manufacturers have no clue how the rest of the LUTs are initialized.

4.3.2 DBF Generation

After initialization of the DBF in the FPGA, the user generates the input-output mapping for the DBF which serves as $f_{complex}$. This can be easily done by traversing all the possible inputs and generating the corresponding outputs. The mapping is stored in the form of Boolean functions. Therefore, until now, the initialized DBF embedded in the FPGA serves as $f_{compact}$ and the direct inputs-outputs mapping forms $f_{complex}$.

5 Security Properties

In this section, we adopt a set of standard statistical tests to analyze the security properties of the digital PUF. Note that since DBF and digital PUF share the same hardware structure, the test results will be the same for DBF. We describe possible statistical attacks and test the resilience of our digital PUF against such attacks. We use the standard digital PUF structure with 64-bit inputs and outputs and 32 levels of substitution. We assume that the digital PUF is initialized randomly.

Table 8 NIST randomness test results on the digital PUF

Statistical test	Avg. success ratio (%)
Frequency	100
Block frequency ($m = 128$)	98.7
Cusum-forward	97.8
Cusum-reverse	97.9
Runs	98.4
Longest runs of ones	97.9
Rank	99.3
Spectral DFT	97.5
Non-overlapping templates ($m = 9$)	97.5
Overlapping templates ($m = 9$)	97.5
Universal	100
Approximate entropy ($m = 8$)	98.1
Rand. excursions ($x = 1$)	98.8
Rand. excursions variant ($x = -1$)	97.6
Serial ($m = 16$)	99.3
Linear complexity ($M = 500$)	98.0

1000 bitstreams of 10,000 bits are provided to each test. Each test passes for p-value $\geq \sigma$, where $\sigma = 0.01$

5.1 Output Randomness

We quantify the output randomness of the digital PUF by applying the industry standard statistical test suite provided by the National Institute of Standards and Technology (NIST). We generate a stream of outputs in the following way: a random seed is used as the primary inputs to the digital PUF after random configuration and the corresponding outputs are generated. In each subsequent clock cycle, the outputs are XORed with the previous inputs to generate the inputs for the next clock cycle. We repeat the process until we collect enough outputs required by the benchmark suite. The results in Table 8 indicate that the output stream of the digital PUF passes the NIST randomness tests with considerably high success ratio.

5.2 Avalanche Effect

In this attack, an adversary attempts to predict the outputs of the digital PUF using the knowledge of outputs for similar inputs. In cryptography, cipher diffusion is achieved if a change in the input by one bit results in a dramatic change in the outputs in an unpredictable manner. This is otherwise known as the avalanche effect. To test this, we measure the hamming distance between two output vectors whose input vector differ by one bit. Ideally, the distribution should be in the form of a

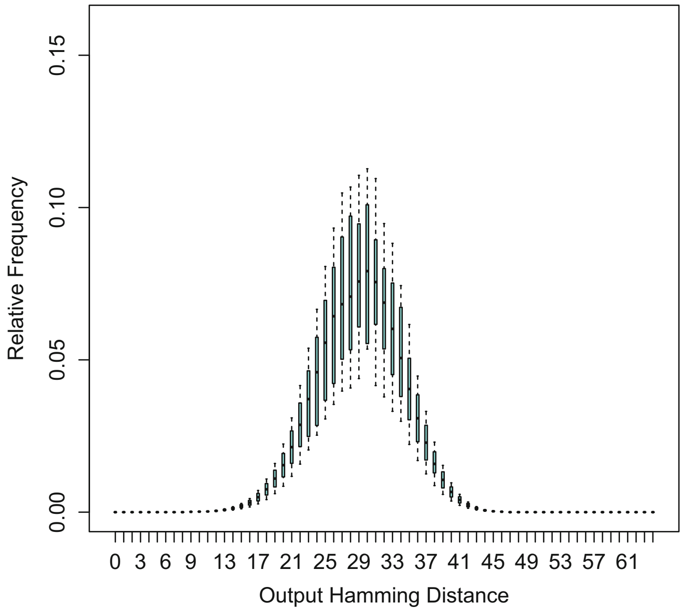


Fig. 11 Distribution of output hamming distances testing the avalanche effect. The error bars depict the max, 0.75 quantile, mean, 0.25 quantile, and min frequencies

binomial distribution with the peak at half of the number of output bits. The result in Fig. 11 shows an almost perfect binomial distribution which indicates our digital PUF satisfies the avalanche criterion and is highly resilient against this type of attack.

5.3 *Input-Based Correlation*

Another type of attack utilizes correlations between individual output bits, O_i, and input bits, I_j, for prediction. The goal in this attack is to predict the conditional probability, $P(O_i = c_1|I_j = c_2)$, where c_1 and c_2 are either 1 or 0. For example, if the attacker observes that output O_i is equal to 1 when the input I_j is 1 a large majority of the time, then he can guess with a high probability that output O_i is 1 when I_j is 1. The ideal situation is when all conditional probabilities are 0.5. Figures 12 and 13 depict the distribution of conditional probabilities, $P(O_i = 1|I_j = 1)$, for the digital PUF. The majority of probabilities cluster around 0.5, thus indicating low potential for prediction.

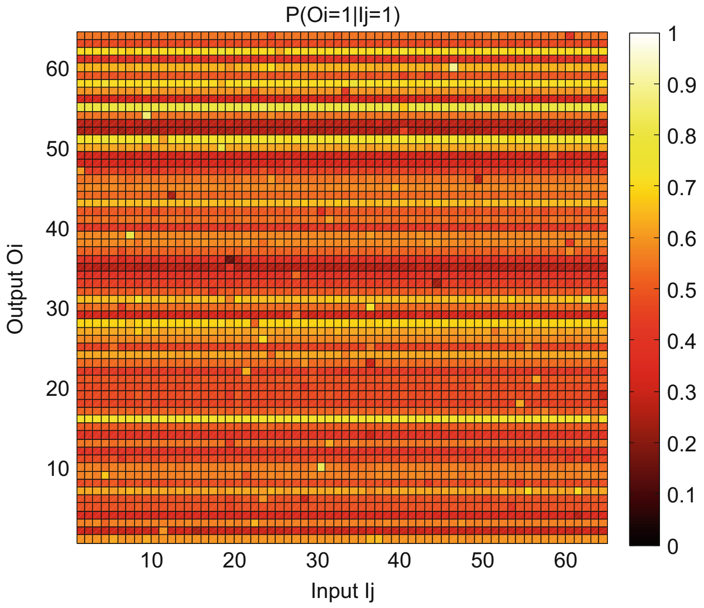


Fig. 12 Colormap of conditional probabilities between output bits O_i and input bits I_j

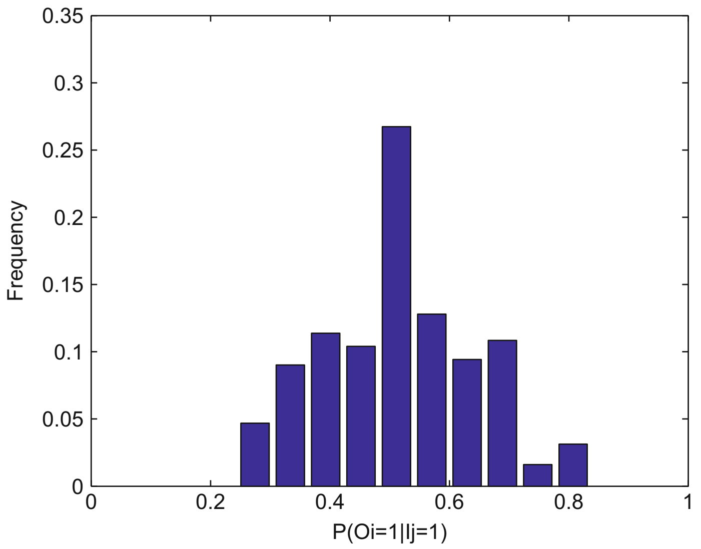


Fig. 13 Distribution histogram of conditional probabilities between output bits O_i and input bits I_j

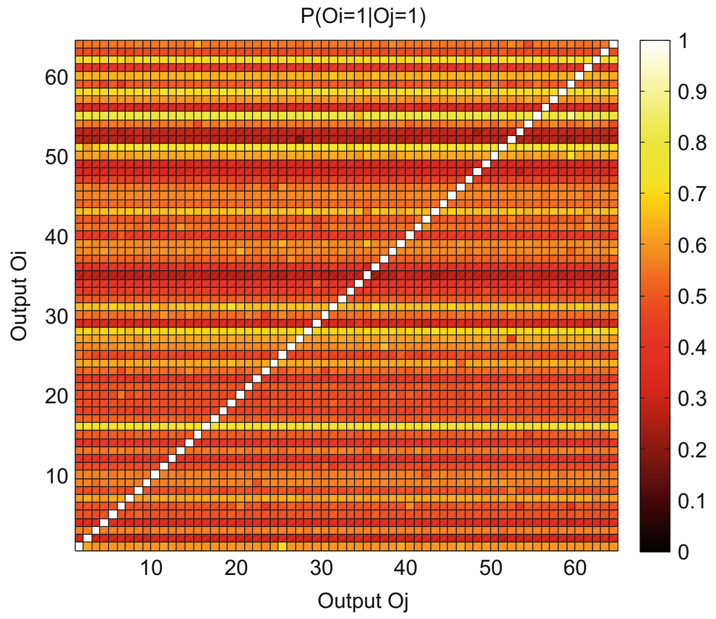


Fig. 14 Colormap of conditional probabilities between output bits O_i and Output bits O_j

5.4 *Output-Based Correlation*

Similar to the previously described attack, this attack attempts to predict an output bit O_i according to the value of a corresponding output bit O_j. In this case, if two output bits have a strong correlation, then the attacker can deduce the output vector through knowledge of a subset of output bits. We present the distribution of conditional probabilities, $P(O_i = 1|O_j = 1)$ in Fig. 14 and the corresponding histogram of the probability distribution Fig. 15 which depicts low potential for prediction based on output to output correlation.

5.5 *Comparisons*

Finally, we briefly compare the statistical test results of our FPGA-based digital PUF with the traditional delay-based PPUF. The results depicted in Table 9 are tested on the 64-input 64-output FPGA-based digital PUF and 64-input 64-output traditional delay PPUF. We conclude that both our PUF and the traditional delay PPUF demonstrate excellent properties regarding output frequency and conditional probability, but only our digital PUF demonstrates an ideal output hamming distance satisfying the avalanche criterion.

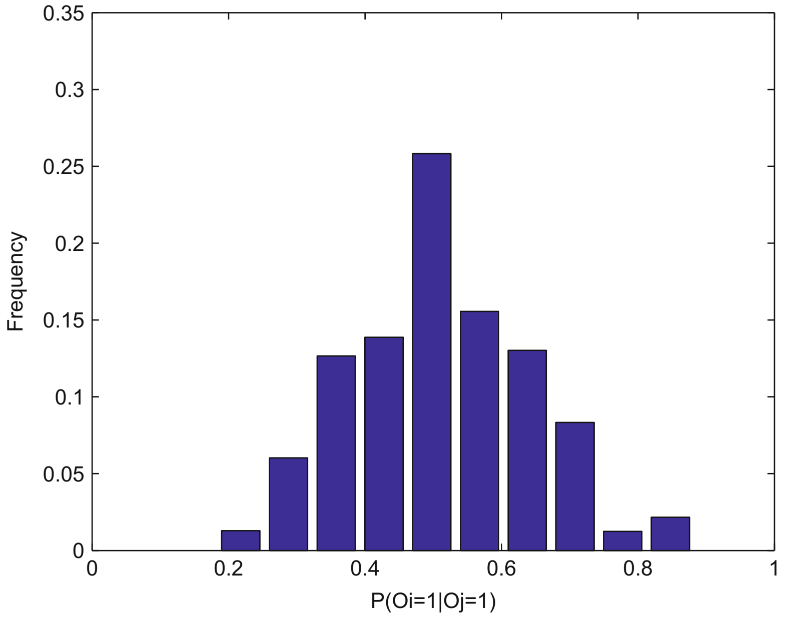


Fig. 15 Distribution histogram of conditional probabilities between output bits O_i and Output bits O_j

Table 9 Statistical test results comparison between the FPGA-based digital PUF and the traditional delay PPUF

	Output frequency	Hamming distance	$P(O_i = 1 \vert I_j = 1)$
Digital PUF	0.5 ± 0.07	30.9 ± 3.6	0.49 ± 0.01
Delay PPUF	0.5 ± 0.09	1.4 ± 0.4	0.52 ± 0.01

The ideal case for output frequency is 0.5, for hamming distance is 32, and for $P(O_i = 1|I_j = 1)$ is 0.5. The results shown in the table are the average values and corresponding standard deviations

6 Protocols

6.1 Public Key Communication

Public key communication is one of the most widely used communication protocols. We use it as an example to explain how digital PUF works as a security primitive in security protocols. The basic setting for this protocol is shown below.

- Private Key—$f_{compact}$, denoted by K_{priv}.
- Public Key—$f_{complex}$, denoted by K_{pub}.
- Alice—the owner of K_{priv}. The party to receive and decrypt messages.
- Bob—the party to send and encrypt messages.
- TTP—trusted third party. The party that administrates K_{pub}.

Before explaining how this protocol works, we need to note that the private key $f_{compact}$ is actually a piece of the digital PUF, in which the hardware implementation of $f_{compact}$ is offered but not detected. By using K_{priv}, given input vectors, the output vectors can be calculated promptly. Meanwhile, the public key is the functions in $f_{complex}$.

Algorithm 1 Public Key Communication

```
1: Bob has a message $m$ to send to Alice, $m$ is in the form of binary vector.
2: Suppose there are $l$ single Boolean sub-functions in $K_{pub}$, all of which have the form of a
   sum of products and the form of a product of sums. In the first round, for the $ith$
   ($i \in \{1, 2, \ldots, l\}$) function $f_i$ in $K_{pub}$, Bob randomly requests either one product from the a
   sum of products or one sum from the a product of sums from TTP. According to that
   product or sum, Bob generates a binary input vector $p_i$, making $r_i = f_i(p_i) = 1$ (case of
   product) or $r_i = f_i(p_i) = 0$ (case of sum). Then Bob concatenates $p_1$ to $p_l$ to generate
   binary vector $P_1$ and concatenates $r_1$ to $r_l$ to generate binary vector $R_1$.
3: Bob repeats step 2 for $N$ rounds, generates $P_j$ and $R_j$ ($j \in \{1, 2, \ldots, N\}$).
4: Bob calculates $E = m \oplus R_1 \oplus R_2 \ldots \oplus R_N$.
5: Bob broadcasts $E$ and all $P_j$ ($j \in \{1, 2, \ldots, N\}$).
6: Alice computes all the sub vector $p_i$ ($i \in \{1, 2, \ldots, l\}$) in each $P_j$ ($j \in \{1, 2, \ldots, N\}$) with
   $K_{priv}$ to find out the corresponding value of $r_i$, then Alice concatenates all the $r_i$ to form
   each vector $R_j$.
7: Alice computes $m = E \oplus R_1 \oplus R_2 \ldots \oplus R_N$ and gets Bob's message.
```

The core idea in Protocol 1 is to take advantage of the calculation time difference of $f_{compact}$ and $f_{complex}$, making only the holder of K_{priv} be the only person who can calculate R_j ($j \in \{1, 2, \ldots, N\}$) in a reasonable amount of time. Another important design is that we use the trusted third party to administrate K_{pub}. Note that whenever Bob wants to send a message to Alice, he only needs to request one product or one sum of each function in K_{pub} from TTP, then he can create the binary vectors to encrypt his messages. This makes use of the sum of products and product of sums; by knowing one product in a sum of products, one input vector that makes the output to be 1 can be deduced and by knowing one sum in a product of sums, one input vector that makes the output 0 can be deduced. This design minimizes the key size as well as the energy for calculation that Bob requires during the encryption of public key communication. However, for an attacker, if he/she wants to find out the right c_x in C, since he/she does not have K_{priv}, he/she can only request all the information of K_{pub} from TTP to simulate. Therefore, for an attacker, the public key size and the calculation scale are not minimized at all, the expected effort of an attacker is to scan half of a sum of products as well as half of a product of sums. In Protocol 1, N rounds of operations are used to boost the calculation expense of $f_{compact}$ and $f_{complex}$ N times simultaneously. As it takes much more time to calculate $f_{complex}$ than $f_{compact}$, after both calculation expense increasing N times, the calculation time for $f_{compact}$ is still trivial but the time for $f_{complex}$ increases significantly. N is a flexible number that can be adjusted according to the size of $f_{compact}$ and $f_{complex}$.

6.1.1 Time Gap

We estimate the decryption time gap between the private key holder and the attacker in Protocol 1. Suppose the private key holder uses an $f_{compact}$ structure with 64 inputs and 32 cycles to implement $f_{compact}$ and the attacker simulates the corresponding $f_{complex}$. For $f_{compact}$, the implementation time is tested on the Spartan-3 XC3S50-5 FPGA and was measured at approximately 239 ns. For the computation of $f_{complex}$, the simulation time is too long and can only be estimated. It is obvious that the simulation time is proportional to the size of the $f_{complex}$. According to Table 1, we find that the size of $f_{complex}$ increases by at least 2.5 times with an increase of 1 iteration and 2 inputs. As a consequence, we can assume the simulation time for $f_{complex}$ also grows similarly. Therefore, by combining the results in Table 3, the simulation time for an $f_{compact}$ structure with 64 inputs and 32 iterations can be estimated at $1.61 \times 10^7 \times 2.5^{22} = 9.15 \times 10^{15}$ (ns). We further assume that the number of rounds N in public key communication is 10^3, therefore, the time for private key calculation is $239 \times 10^3 = 2.39 * 10^5$ (ns) while the time for public key calculation is $9.15 \times 10^{15} \times 10^3 = 9.15 * 10^{18}$ (ns) ≈ 290 years, which is not acceptable. Note that N can be designed to be smaller with a larger size public key.

6.1.2 Performance Comparisons

We estimate the performance of the digital PUF based public key communication protocol and compare it with other cryptographic methods. For decryption, suppose the digital PUF uses a 64 input $f_{compact}$ structure, as a result, the area of 64 LUTs is required. The static power value of the Xilinx FPGA is approximately 24 $\mu W/CLB$. Suppose each CLB contains 4 LUTs, then 16 CLBs are required with a total power of approximately 384 μW. According to the synthesis result, the maximum delay of the $f_{compact}$ structure circuit is 7.47 ns and 32 cycles are needed to compute the outputs. We repeat step 2 in Protocol 1 N= 10^3 times, so that the total clock cycle that decryption requires is 3.2×10^4. For encryption, according to the protocol, the public key user only needs to calculate one product or one sum of each function in $f_{complex}$ in step 2 of protocol 1, compared to the decryption part, the expense of the encryption can be neglected. We therefore have an energy consumption estimation for digital PUF based public key communication through Eq. (2) .

$$Energy = Power \times ClockCycle\ \# \times MaximumDelay \tag{5}$$

Table 10 shows the comparisons for DBF based public key communication with traditional block cyphers and RSA with respect to area, delay, and energy in FPGAs. The DBF in the table utilizes one $f_{compact}$ structure with 64 primary inputs and 32 iterations, and we suppose that the number of rounds N in the protocol is 10^3. We can obviously conclude from the table that DBF, as a security primitive, owns the implementation of ultra low energy that is competitive with the traditional security key block cyphers and outperforms the RSA with at least three orders of magnitude.

Table 10 Comparisons for DBF based cryptography with the traditional block cyphers and RSA

Design	Flip Flops	LUTs	(Area Slices)	Maximum Delay (ns)	Clock Cycles	Energy (μJ)	Block Size (bits)	Throughput (Mbps) at f_{max}	Throughput/Energy (Mbps/μJ)	Device
Present[25]	114	159	117	8.78	256	3.16×10^{-3}	64	28.46	9.01×10^{3}	xc3s50−5
HIGHT[25]	25	132	91	6.12	160	1.07×10^{-3}	64	65.48	6.12×10^{4}	xc3s50−5
AES[25]	338	531	393	14.21	534	3.58×10^{-2}	128	16.86	4.71×10^{2}	xc3s50−5
RSA[26]	1870	2811	1553	7.62×10^{3}	907	128.80	–	0.15	1.16×10^{-3}	xc3s500e
RSA radix-2[26]	7564	11496	6282	8.21×10^{3}	1058	654.80	–	0.12	1.83×10^{-4}	xc2v6000
RSA radix-4[26]	9944	14907	8328	4.23×10^{3}	560	236.73	–	0.43	1.82×10^{-3}	xc2v6000
DBF	64	64	32	7.47	32000	9.18×10^{-2}	–	267.74	2.92×10^{3}	xc3s50−5

The results for Present, HIGHT and AES are cited from [25], the results for RSA are the parts of multiplication modular and are cited from [26], the results for DBFs are tested on the Spartan-3 XC3S50-5 FPGA and generated by the Xilinx ISE Design Suite 14.3

6.2 Remote Trust

When using a data centre to adopt remote computations, trust plays a very important role. On one hand, users need to authenticate the data centre, on the other hand, users also want to monitor the flow of their requested calculations to ensure the processing of the data is being carried out correctly. Digital PUFs provide an ultra low-energy and easy solution to authenticate the data centre as well as monitor the calculation flow. The basic approach is to use a hash tree.

The calculation flow shown in Fig. 16 is an example of how a data centre processes calculations to generate the outputs. Four basic calculations ($+, -, \times, \div$) are adopted as operation nodes in the flow. The data centre is required to randomly choose n places to "cut" the calculation flow, and generate n corresponding intermediate results. The calculation expense between the adjacent intermediate results is trivial. A hash tree is constructed based on the n intermediate results. Figure 17 shows an example of a hash tree with 4 leaf nodes, where each node represents a intermediate result from the data flow. A binary tree structure is generated in which

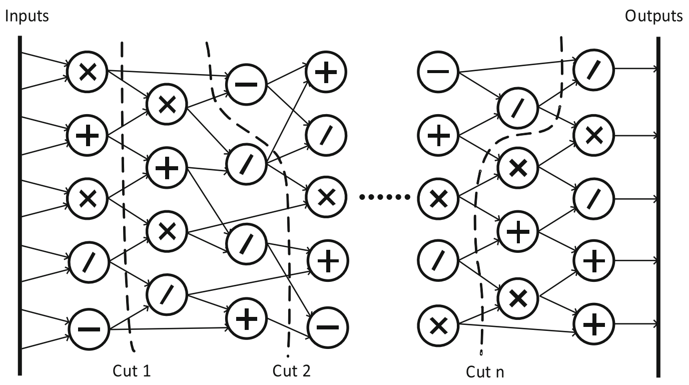


Fig. 16 Example calculation flow and corresponding cuts. Each node in the graph represents a basic operation (e.g. $+, -, \times, \div$) or even blocks of operations (e.g. if-else, while, functions). Cut 1 to cut n represent random cuts in the calculation flow which can be thought of as intermediate results or states of the procedure

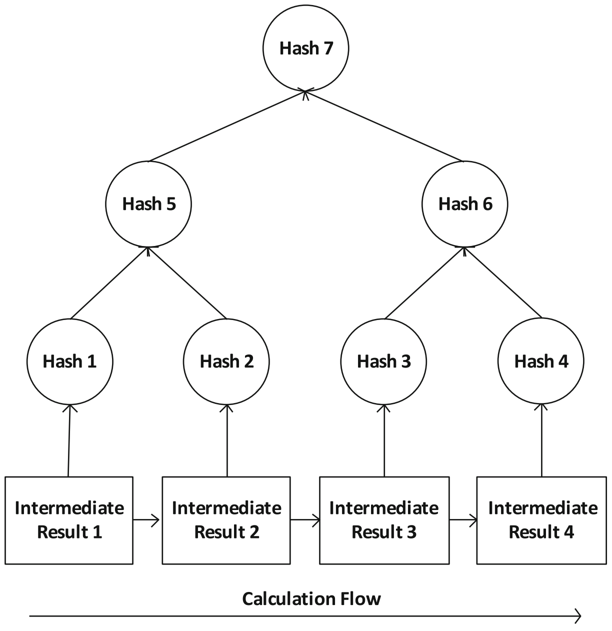


Fig. 17 An example of a Hash tree for our *low* overhead remote trust protocol. The intermediate results at each cut in the calculation flow (e.g. cut i, $i \in \{1, 2, \ldots, n\}$ in Fig. 16) are hashed as leaf nodes. The *arrow* shows the direction of calculation flow

every non-leaf node is the hash of the its two children nodes. Depends on the scale of the calculation, a hash tree may include millions of intermediate results, and the leaf nodes from left to right coordinate with the sequence of the intermediated results generated in the data flow.

To apply a digital PUF on the structure of hash tree, we use the following basic settings.

– Use DBF as the hash function.
– Only the data centre has the digital PUF, which is the hardware implementation of DBF form $f_{compact}$.
– DBF form $f_{complex}$ is public.

Based on these settings, every time a client wants to monitor the calculation, he/she randomly chooses a leaf node of the hash tree and requests the data centre to offer the "corresponding hashed results" to calculate the path from the leaf node to the top node. For example, in Fig. 17, the top value of the hash tree can be verified by iteratively hashing intermediate result 2 with the results in hash 1 and hash 6. In this case, the results in hash 1 and hash 6 are the corresponding hashed results for intermediate result 2. After acquiring the "corresponding hashed results", the client hashes the chosen intermediate result and confirms the rightness of the hashing path afterwards. By repeating this procedure the consistence of the whole hash tree is checked. Another verification that a client can do is to check the calculation between the adjacent intermediate results, e.g., intermediate result 2 and 3. By knowing intermediate result 2, the client can follow the calculation flow to calculate and confirm the intermediate result 3. As the calculation expense between adjacent results is trivial, the client can easily verify the consistence of the adjacent nodes. Therefore, the client can detect the hash tree both horizontally and vertically. Due to the randomness of every request, only the party that has the whole structure of the hash tree can respond to all the requests correctly, therefore, the calculation flow is monitored and detected.

Monitoring of the calculation flow is used to verify whether the data centre is processing the flow correctly. Now suppose the data centre is not honest and mixes some wrong or irrelevant calculations in the flow. Figure 18 shows the possibility that the data centre can pass all the adjacent nodes tests from the client with only some percentage of right calculations in the flow. The results show that with a linear decrease of the proportion of right calculation, the passing ratio drops exponentially. For example, when the right calculation proportion reaches 1 % in the case of 10 test pairs, the passing ratio drops to be 10^{-20}, which is ridiculously small.

Note that since the data centre needs to hash millions of intermediate results during the calculation flow, to complete the hashes in a reasonable short time, $f_{compact}$ must be used. Any unauthenticated party who only has $f_{complex}$ suffers a long time of hashing. As an example, suppose we use the $f_{compact}$ structure with 20 inputs and 10 iterations in remote trust. Again, we emphasize that we have flexibility in choosing $f_{compact}$ structure's size for different applications. Suppose that the number of intermediate results is 10^6 and the client makes 10 hash operations. According to Table 9, the calculation time for the data centre is $77.2 \times 10^6\,\text{ns} = 0.0772\,\text{s}$ and the calculation time for the client is $1.61 \times 10^7 \times 10\,\text{ns} = 0.161\,\text{s}$. However, the calculation time for an attacker reaches $1.61 \times 10^7 \times 10^6\,\text{ns} = 1.61 \times 10^4\,\text{s} = 4.47\,\text{h}$. As a consequence, only the party that can offer requested hash results correctly in a reasonably short period of time will be certified by the client. In this way, the only party that can be authenticated is the data centre with the digital PUF.

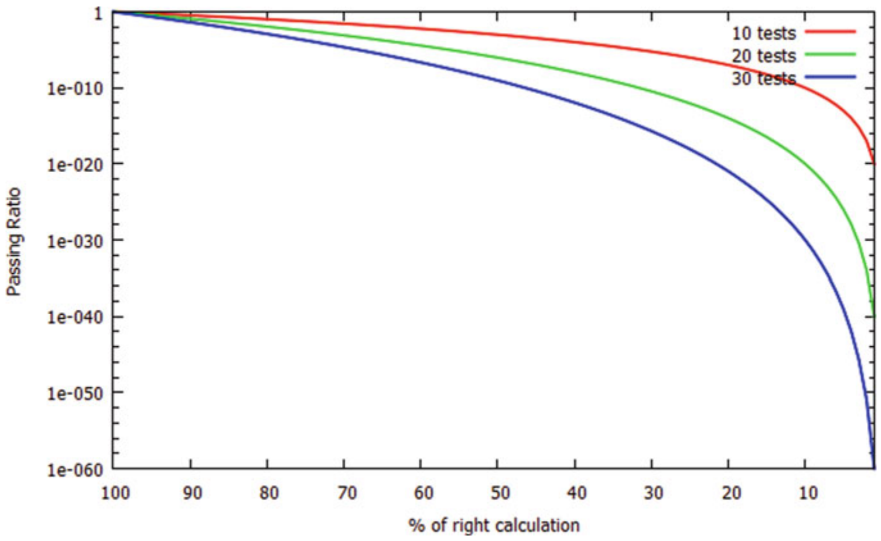


Fig. 18 Passing ratio with the proportion of *right* calculation. The three curves respectively shows the passing ratio under the circumstance that the client requests 10, 20, and 30 pairs of adjacent nodes to test

7 Conclusion

We have presented the concept of DBF and digital PUF which resolve two essential problems in traditional analog PUF. The first problem is stability. Analog PUF is unstable in the same sense that analog system is unstable. Digital PUF resolves this by leveraging a stable delay-based PUF for initializing its connected network of LUTs of digital bimodal functions (DBFs). The stability in the delay-based PUF is ensured by selecting challenges that have a delay ratio of at least 10 % which ensures that the output is always stable for temperatures ranging from 250 to 400 K. This process guarantees the system to be both unclonable and stable. The second problem of analog PUF is hard to be integrated with digital logic. Digital PUF resolves this problem by employing completely digital system.

Digital PUF is built based on the structure of DBF. DBF has two forms of functions, among which one form is fast and compact, the other form is slow and complex. We proposed the architecture of DBF on FPGA and analyzed the security properties. The security analysis indicates that DBF can pass all benchmark tests from the NIST randomness suite, as well as the avalanche criterion.

Finally, two security protocols are demonstrated, respectively security protocols and remote trust. For security protocols, a fast speed, low overhead public key communication protocol is proposed. It employs the huge computation gap between the two forms of functions in DBF. We have also demonstrated the application of the digital PUFs in a remote trust protocol in which both communicating parties experience very low overhead in terms of both time and energy.

References

1. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. National Institute of Standards and Technology (NIST) Special Publication 800-22, Rev. 1a, April 2010
2. Shannon, C.E.: Communication theory of secrecy systems. Bell Syst. Tech. J. **28**(4), 656–715 (1949)
3. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science **297**(5589), 2026–2030 (2002)
4. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In:Proceedings of the 9th ACM conference on computer and communications security, pp. 148–160. ACM, New York (2002)
5. Guajardo, J., Kumar, S., Schrijen, G., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Paillier P., Verbauwhede I. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2007, pp. 63–80. Springer, Berlin/Heidelberg (2007)
6. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Proceedings of the 44th annual Design Automation Conference (DAC 2007), pp. 9–14. ACM, New York (2007)
7. Lee, J.W., et al.: A technique to build a secret key in integrated circuits for identification and authentication applications. In:Symposium on VLSI Circuits, pp. 176–179 (2004)
8. Devadas, S., et al.: Design and implementation of PUF-based "Unclonable" RFID ICs for anti-counterfeiting and security applications. In: IEEE International Conference on RFID, pp. 58–64. IEEE (2008)
9. Simpson, E., Schaumont, P.: Offline hardware/software authentication for reconfigurable platforms. In: Goubin, L., Matsui M. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2006, pp. 311–323. Springer, Berlin/Heidelberg (2006)
10. Alkabani, Y., Koushanfar, F.: Active hardware metering for intellectual property protection and security. In: USENIX Security Symposium, pp. 291–306 (2007)
11. Potkonjak, M., Meguerdichian, S., Wong, J.L.: Trusted sensors and remote sensing. In: IEEE Sensors, pp. 1104–1107 (2010)
12. Wendt, J.B., Potkonjak, M.: Nanotechnology-based trusted remote sensing. In: IEEE Sensors, pp. 1213–1216 (2011)
13. Xu, T., Potkonjak, M.: Lightweight digital hardware random number generators. In: IEEE Sensors, pp. 1–4 (2013)
14. Suh, G.E., et al.: Design and implementation of the AEGIS single-chip secure processor using physical random functions. In: ACM SIGARCH Computer Architecture News, vol. 33, pp. 25–36 (2005)
15. Beckmann, N., Potkonjak, M.: Hardware-Based Public-Key Cryptography with Public Physically Unclonable Functions. In: Information Hiding: 11th International Workshop, pp. 206–220, Darmstadt, Germany (2009)
16. Rührmair, U.: SIMPL systems, or: can we design cryptographic hardware without secret key information? In: Černá I., et al. (eds.) SOFSEM 2011: Theory and Practice of Computer Science, pp. 26–45. Springer, Berlin/Heidelberg (2011)
17. Meguerdichian, S., Potkonjak, M.: Matched public PUF: ultra low energy security platform. In:IEEE/ACM ISLPED, pp. 45–50 (2011)
18. Xu, T., Wendt, J.B., Potkonjak, M.: Security of IoT Systems: Design Challenges and Opportunities. In: IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 417–423 (2014)
19. Potkonjak, M., Meguerdichian, S., Nahapetian, A., Wei, S.: Differential Public, Physically Unclonable Functions: Architecture and Applications. In: ACM/IEEE Design Automation Conference, pp. 242–247 (2011)
20. Alam, M.A., Mahapatra, S.: A comprehensive model of PMOS NBTI degradation. Microelectron. Reliab. **45**(1), 71–81 (2005)

21. Xu, T., Wendt, J.B., Potkonjak, M.: Digital Bimodal Function: An Ultra-Low Energy Security Primitive. In: International Symposium on Low Power Electronics and Design (ISLPED), pp. 292–297 (2013)
22. Xu, T., Potkonjak, M.: Robust and Flexible FPGA-based Digital PUF. In: International Conference on Field Programmable Logic and Applications, pp. 1–6, Sept 2014
23. Xu, T., Wendt, J.B., Potkonjak, M.: Secure Remote Sensing and Communication using Digital PUFs. In: Symposium on Architectures for Networking and Communications Systems, pp.1–12, Oct 2014
24. Huang, W., Ghosh, S., Velusamy, S., Sankaranarayanan, K., Skadron, K., Stan, M.R.: Hotspot: A compact thermal modeling methodology for early-stage VLSI design. IEEE Trans. VLSI Syst. **14**(5), 501–513 (2006)
25. Yalla, P., Kaps, J.-P.: Lightweight cryptography for FPGAs. In: International Conference on Reconfigurable Computing and FPGAs, 2009. ReConFig'09. IEEE (2009)
26. Oksuzoglu, E., Savas, E.: Parametric, secure and compact implementation of RSA on FPGA. In: International Conference on Reconfigurable Computing and FPGAs, 2008. ReConFig'08. IEEE (2008)

Residue Number Systems in Cryptography: Design, Challenges, Robustness

Dimitris Schinianakis and Thanos Stouraitis

Abstract As conventional arithmetic solutions have improved at a fine-grain level, researchers have turned their attention to alternative number system representations in an effort to further boost up cryptosystem performance. The ancient Residue Number System (RNS) has emerged as a key-player in this endeavor. This chapter attempts to highlight important concepts of residue arithmetic and new RNS applications in modern cryptography are presented in a systematic and holistic manner. Progressing from algorithm and complexity analysis to state-of-the-art hardware implementations and useful cryptanalytic properties, the prospective reader is acquainted with most of the implications and challenges of this emerging field, while open research points are also highlighted.

1 History

The Residue Number System (RNS) is a data representation system that allows representing an integer number as a set of smaller numbers. RNS was originally described in terms of a riddle by Nicomachus of Gerasa (60-120 CE) in his book "*Introduction to Arithmetic*". Later, the problem was re-described by Sun Tzu in a fifth century book entitled "*Sunzi Suanjing*"(The Mathematical Classic of Sunzi). Sun Tzu described the problem of determining the value of a number x by knowing only the residues after its division with a predefined number of divisors.

Sun Tzu devised a methodology for manipulating remainders of an integer after division by 3, 5, and 7. This contribution is commemorated today as the Chinese Remainder Theorem (CRT). This theorem, as well as the theory of residue arithmetic, were further explored and new advances were presented in the nineteenth century by Carl Friedrich Gauss in his famous "*Disquisitiones Arithmetical*" [57].

D. Schinianakis
Department of Electrical and Computer Engineering, University of Patras, Patras, Greece
e-mail: dsxoiniana@gmail.com

T. Stouraitis (✉)
Department of Electrical and Computer Engineering, University of Patras, Rio 26110, Greece
e-mail: thanos@upatras.gr

© Springer International Publishing Switzerland 2016
C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_4

The first attempt to explore some of the remarkable RNS properties was made by D. H. Lenhmet who, in 1932, built a special-purpose machine he named "photo-electric sieve". This device factored Mersenne numbers. Then, in the mid-1950s, two Czech researchers, Svadoba and Valach, experimented with a hardwired, small-moduli RNS device, which was used to study error codes. The same idea was conceived by Aiken and Garner around the same period [57]. Perhaps the most remarkable advance from the late 1950s to mid-1960s, was the research work of Szabo and Tanaka at Lockheed. They worked on a special-purpose correlator, while at the same period a research group from RCA designed a general-purpose machine. These efforts however did not meet much success, since the technology in 1950s and 1960s could not cope with the unique demands of RNS arithmetic. Perhaps the most important result of this research work is a comprehensive text by Szabo and Tanaka which survived only one printing [54, 57].

Since the mid-1970s, however, technology and theory slowly started converging. More than 150 papers had been produced from mid-1970s until mid-1980s, while the first patents and books on RNS also found their way during this period. Initially, the main application area of RNS was *Digital Signal Processing* (DSP). Huang built and tested a two-dimensional, RNS, matched filter capable of 20M operations per second [20], while Smith at Martin-Marietta developed a high-speed FFT in the RNS [52]. Jullien reported an RNS comb filter in [22] while Taylor published on RNS systems with VLSI hardware [56]. Numerous works on RNS DSP applications are still being published. In the 1990s, Taylor et al. presented the so-called "Gauss-machine", a DSP processor with high RNS content [55], while Claudio et al. proposed some fast combinatorial RNS arithmetic modules for DSP use [11]. Ramirez et al. proposed in 2001 an RNS implementation for the Discrete Wavelet Transformation (DWT) [40].

While research on the properties and DSP design implications of RNS arithmetic bloomed in the 1980s and 1990s, cryptography was just starting to make its way through the research community, especially when it comes to digital design. The reasons are quite simple; during the 1980s, the need for securing digital data was minimal and at the same time technology was struggling to support operations on data of very large word length, as it is required by public-key cryptosystems [42]. This discouraged researchers from exploring RNS for use in cryptography and it was not until the mid-1990s when the work of Posch and Posch [38, 39] set the basis for a new research field. This work did not explicitly refer to cryptosystem design. It proposed a method to embed RNS arithmetic in modular multiplication which, as we will see later, it is the most time- and area-consuming operation that determines to a large extent the overall cryptosystem performance.

In modern cryptosystem design, the main RNS application is related to Montgomery modular multiplication [35], as it is well-suited to RNS arithmetic, since it avoids hard divisions. The work by Kawamura et al. in the 2000s is considered as a cornerstone for the implementation of competitive RNS-based crypto-processors [24, 37]. Since mid-2000s a considerable amount of research on RNS Montgomery multiplication and efficient cryptosystem design has been published; the first exploitation of RNS arithmetic in Elliptic Curve Cryptography (ECC) was presented in [45], while RNS was also applied in RSA cryptography [5, 14, 15, 48]. Studies on the underlying security of RNS Montgomery multipliers have also appeared in the literature, contributing significantly to the expansion of RNS-based cryptosystems and their establishment as a viable design option [2, 47].

This 2000 year old system is now turning to a competitive key-player next to more traditional options for performing modular operations on data of large word length. In the following we will analyze the design challenges of RNS arithmetic in cryptographic systems, present design methodologies and important algorithmic advances, and discuss useful side effects of RNS exploitation in cryptosystem security.

2 Cryptographic Algorithms of Interest

In the following, we focus on some of the most widely used public-key cryptography algorithms, which pose extra difficulties in their implementation due to their data-intensive character, namely the RSA algorithm [42] and Elliptic Curve Cryptography (ECC) [26, 33]. Regarding the underlying arithmetic, we assume that the reader is acquainted with basic field operations in Galois fields of the forms $GF(p)$ and $GF(2^n)$. In $GF(p)$, elements are all integers in $[0, p-1]$ and arithmetic is performed modulo p, where p a prime. Field elements in $GF(2^n)$ are polynomials represented as binary vectors of dimension n, relative to a given polynomial basis $(1, \alpha, \alpha^2, \ldots, \alpha^{n-1})$, where α is a root of an irreducible polynomial $p(x)$, of degree n, over $GF(2)$. The field is then realized as $GF(2)[x]/(p)$ and the arithmetic is that of polynomials of degree at most $n-1$, modulo p [7, 10, 12, 25]. The addition of two polynomials a and b in $GF(2^n)$ is performed by adding the polynomials, with their coefficients added in $GF(2)$, i.e., modulo 2. This is equivalent to a bit-wise XOR operation on the vectors a and b.

2.1 The RSA Cryptosystem

RSA is an algorithm for public-key cryptography that is based on the presumably difficult mathematical problem of factoring large integers. RSA stands for the initials of Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described the algorithm in 1977 [42]. Clifford Cocks, an English mathematician, had developed an equivalent system in 1973, but it was not classified until 1997.

In the RSA cryptosystem, the public and private keys are generated by two distinct prime numbers, p and q. We calculate the public modulus $N = pq$ and the quantity $\varphi(N) = (p-1)(q-1)$, where φ is the Euler's totient function. We choose $e \in \mathbb{Z}$, a co-prime to $\varphi(N)$, and we compute $d = e^{-1} \mod \varphi(N)$. The public key is the pair (N, e) and the private key is d. The primes p, q are also kept secret. The public and private keys are referred to as public and secret exponent, respectively. The encryption of a message M is defined by

$$C = M^e \mod N \tag{1}$$

and decryption by

$$M = C^d \mod N. \tag{2}$$

2.1.1 RSA-CRT Algorithm

The security of RSA depends on the key size. With large keys varying from 1024 bits, appropriate for protecting data through the year 2015, to 2048 bits, appropriate through the year 2035 [23], it is apparent that efficient arithmetic operations on large operands are crucial for optimal RSA implementations.

A step towards efficiency was the introduction of the Chinese Remainder Theorem (CRT) to the RSA protocol, namely the RSA-CRT [28, 29]. In RSA-CRT, the digital signature operation $S = M^d \mod N$ is split in two operations $S_p = M^{d_p} \mod p$ and $S_q = M^{d_q} \mod q$, where $d_p = d \mod (p-1)$ and $d_q = d \mod (q-1)$. CRT ensures that the combination of S_p and S_q produces signature S as

$$S = S_q + \left[\left(S_p - S_q\right) \cdot \left(q^{-1} \mod p\right) \mod p\right] \cdot q, \tag{3}$$

denoted from now on as $S = CRT(S_p, S_q)$ [25], with a speedup of operations of approximately four times [28, 29].

2.2 *Elliptic Curve Cryptography*

Elliptic Curve Cryptography (ECC), presented by Koblitz [26] and Miller [33] independently in 1985, has withstood a large number of attacks and has evolved significantly, so that it is considered nowadays a mature public-key cryptosystem. Extensive research work is currently focusing on the underlying mathematics, security, and its efficient implementations.

By exploiting the Elliptic Curve Discrete Logarithm Problem (ECDLP), ECC offers the highest strength per bit and the smallest key size, when compared to other public-key cryptosystems. ECDLP states that given two points P, Q, on an elliptic curve, such that $Q = [k]P$, it is computationally infeasible to calculate $[k]$ [7].

Although elliptic curves can be defined on a variety of different fields, only finite fields are employed in cryptography. Among them, prime fields, $GF(p)$, and binary extension fields, $GF(2^n)$, are considered to be the ones that offer the most efficient and secure implementations [7].

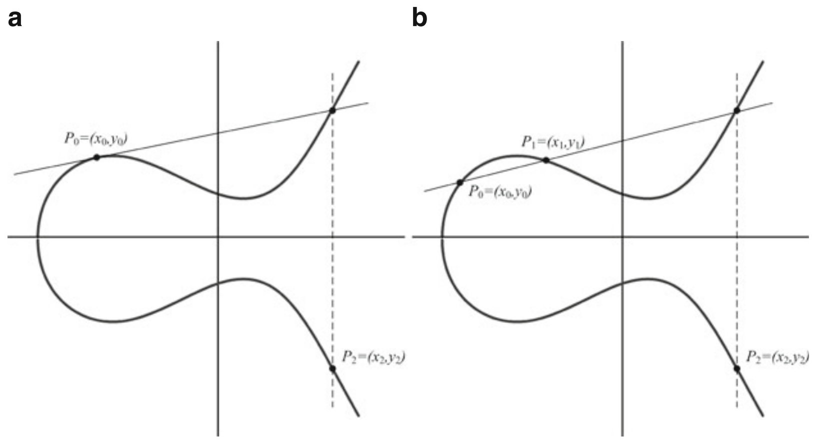


Fig. 1 Operations on elliptic curves. (**a**) Point doubling. (**b**) Point addition

2.2.1 Elliptic Curves Over *GF*(*p*)

An elliptic curve $\mathcal{E}$ over $GF(p)$ is defined by an equation of the form

$$y^2 = x^3 + ax + b, \tag{4}$$

where $a, b \in GF(p)$ and $4a^3 + 27b^2 \neq 0 (\mathrm{mod}\, p)$, together with a special point $\mathcal{O}$, called the *point at infinity*. The set $\mathcal{E}(GF(p))$ consists of all points $(x, y), x, y \in GF(p)$, that satisfy (4), together with $\mathcal{O}$. Addition of two points on an elliptic curve can be defined by the group law. Together with the addition operation, the set of points $\mathcal{E}(GF(p))$ forms a group, with $\mathcal{O}$ serving as its identity element. It is this group that is used in the construction of elliptic curve cryptosystems. The special case of adding a point to itself is called a point doubling.

Examples of point addition and point doubling are depicted in Fig. 1. The double of a point P_0 is obtained by taking the tangent line on P_0 until a second intersection point on the curve is found (there is always a second point due to the form of (4)). The mirror point of this second intersection with respect to the x-axis is $2P_0$. Similarly, to add two points P_0, P_1, a third intersecting point is found by the line that connects P_0, P_1. The mirror point of the third intersection point is $P_2 = P_0 + P_1$.

Let $P_0 = (x_0, y_0), P_1 = (x_1, y_1) \neq \mathcal{O}$ and $P_0 \neq -P_1$. The coordinates of their sum, $P_2(x_2, y_2)$ are

$$P_2 = P_0 + P_1 = \begin{cases} x_2 = \lambda^2 - x_0 - x_1 \\ y_2 = (x_0 - x_2)\lambda - y_0, \end{cases} \tag{5}$$

where $\lambda = \frac{y_1 - y_0}{x_1 - x_0}$. The double of a point is given by

$$P_2 = 2P_0 = \begin{cases} x_2 = \lambda^2 - 2x_0 \\ y_2 = (x_0 - x_2)\lambda - y_0, \end{cases} \tag{6}$$

where $\lambda = \frac{3x_0^2 + a}{2y_0}$.

From (5), (6), it is apparent that in order to perform an addition or a doubling of a point in *affine* representation, one needs to compute the inverse of an element in $GF(p)$, which is a time consuming operation [7]. In order to avoid inversions, the use of *projective coordinates* of the Elliptic Curve (EC) points has been proposed [7]. Given a point $P = (x, y)$ in affine coordinates, the projective coordinates $P = (X, Y, Z)$ are given by

$$X = x;\ Y = y;\ Z = 1. \tag{7}$$

There are various projective coordinate representations that lead to more efficient implementations than using the one in (7). Jacobian coordinates are an example of such a representation [7]. Using Jacobian coordinates, the affine representation of an EC point is given by

$$x = \frac{X}{Z^2};\ y = \frac{Y}{Z^3}. \tag{8}$$

while the point at infinity is given by $\mathcal{O} = (0, 0, 1)$.

Using the representation in (8), (4) rewrites to

$$\mathcal{E}\,(GF(p)):\ Y^2 = X^3 + aXZ^4 + bZ^6. \tag{9}$$

Let $P_0 = (X_0, Y_0, Z_0),\ P_1 = (X_1, Y_1, Z_1) \in \mathcal{E}(GF(p))$. The sum $P_2 = (X_2, Y_2, Z_2) = P_0 + P_1 \in \mathcal{E}(GF(p))$ can be computed as follows.
If $P_0 = P_1$ then

$$P_2 = 2P_1 = \begin{cases} X_2 = M^2 - 2S \\ Y_2 = M(S - X_2) - T, \\ Z_2 = 2Y_1 Z_1 \end{cases} \tag{10}$$

where $M = 3X_1^2 + aZ_1^4, S = 4X_1Y_1^2$ and $T = 8Y_1^4$. On the other hand, if $P_0 \neq P_1$, then

$$P_2 = P_0 + P_1 = \begin{cases} X_2 = R^2 - TW^2 \\ 2Y_2 = VR - MW^3, \\ Z_2 = Z_0 Z_1 W \end{cases} \tag{11}$$

where $R = Y_0Z_1^3 - Y_1Z_0^3, T = X_0Z_1^2 + X_1Z_0^2, W = X_0Z_1^2 - X_1Z_0^2, M = Y_0Z_1^3 + Y_1Z_0^3$, and $V = TW^2 - 2X_2$.

Elliptic curves over $GF(2^n)$ are defined in a similar manner; it is, however, out of scope of this chapter to emphasize further [7].

2.2.2 Point Multiplication

With the operations of point doubling and point addition available, the next step is to implement the scalar point multiplication, which is the most important operation in ECC. Among various options [7], the *binary method algorithm* is frequently chosen, because it is easy to implement and minimizes memory requirements. The binary method algorithm [7] is based on the binary expansion of the scalar $[k]$, as follows.

Algorithm 1 Binary method for EC point multiplication

```
Input: A point $P$, an $l$-bit integer $k = \sum_0^{l-1} k_j 2^j$
Output: $Q = [k]P$
1: $Q \longleftarrow \mathcal{O}$
2: for $j = l - 1$ to 0 do
3:     $Q \longleftarrow [2]Q$
4:     if $k_j = 1$ then
5:         $Q \longleftarrow Q + P$
6:     end if
7: end for
8: return $Q$
```

The binary method requires $l - 1$ point doublings and $W - 1$ point additions, where l is the length and W the Hamming weight of the binary expansion of k. For any positive integer k, the notation $[k]$ is used to denote the *multiplication-by-k* map from the curve to itself. The notation $[k]$ is extended to $k \leq 0$ by defining $[0]P = \mathcal{O}$, and $[-k]P = -([k]P)$. Other methods based on various representations for the scalar $[k]$ include window-based algorithms, signed-digit representations, NAF representations, etc [7], which are out of scope of this chapter to analyze further.

3 Residue Arithmetic

RNS consists of a set of L, pair-wise relatively prime integers $(m_1, m_2, \ldots, m_L)$, called the *moduli*. The set $\mathcal{M} = (m_1, m_2, \ldots, m_L)$ is called the RNS *base*. Each RNS base is associated with a corresponding range, computed as

$$M = \prod_{i=1}^{L} m_i. \tag{12}$$

Any integer $Z \in [0, M-1]$ has a unique RNS representation $Z_{\mathcal{M}} = (z_1, z_2, \ldots, z_L) = (\langle Z\rangle_{m_1}, \langle Z\rangle_{m_2}, \ldots, \langle Z\rangle_{m_L})$, where $\langle Z\rangle_{m_i}$ denotes the operation $Z \mod m_i$.

Assuming two integers X, Y, in RNS format, i.e., $X_{\mathcal{M}} = (x_1, x_2, \ldots, x_L)$ and $Y_{\mathcal{M}} = (y_1, y_2, \ldots, y_L)$, one can perform the operations $\otimes \in (+, -, *)$, in parallel, as

$$X_{\mathcal{M}} \otimes Y_{\mathcal{M}} = (\langle x_1 \otimes y_1\rangle_{m_1}, \langle x_2 \otimes y_2\rangle_{m_2}, \ldots, \langle x_L \otimes y_L\rangle_{m_L}) . \tag{13}$$

It is important to note that the ith RNS digit of the result is defined in terms of $\langle x_i \otimes y_i\rangle_{m_i}$ only, meaning that no carry information needs to be communicated between residue digits. Since there is no carry propagation from one RNS channel to another, the carry-related overhead that plagues traditional, weighted number systems is eliminated. The result is high-speed, parallel operations. For a signed number system, any integer $Z \in (-M/2, M/2)$ also has an RNS L-tuple representation $z_i = Z \mod m_i$, if $Z > 0$, and $(M - |Z|) \mod m_i$, otherwise.

In order to build up a competitive RNS system it is essential to build efficient blocks that handle the operations $\langle x_i \otimes y_i\rangle_{m_i}$. Next, we elaborate on particular modular operations.

3.1 Division, Magnitude Comparison, Sign Detection

Although addition, subtraction, and multiplication are efficiently supported by RNS arithmetic, unfortunately this is not the case for other operations, like division [19, 62], magnitude comparison [53], or sign detection [58]. Division is basically a combination of nested subtractions and magnitude comparisons, so it is inefficient to perform in RNS, not to mention that RNS is an integer system and so it is not closed under division.

On the other hand, in a weighted number system, we can compare numbers in a bit-by-bit manner, starting from the MSB. Sign detection is a special case of magnitude comparison of a number with 0; also a difficult operation in RNS. For example, for the base $\mathcal{M} = (3, 4, 5)$, one can not tell by simply comparing the RNS digits if $7 \xrightarrow{RNS} (1, 3, 2) < 10 \xrightarrow{RNS} (1, 2, 0)$. Fortunately, such operations are not that common in cryptography. Therefore, we won't elaborate on the implications and peculiarities of RNS regarding these operations.

3.2 Residue-to-Decimal Conversion

Converting an integer from RNS to binary representation is a significant operation in RNS applications. Its efficiency determines to a great extent the efficiency of the overall RNS processor [45]. For the rest of this discussion let us assume, that each

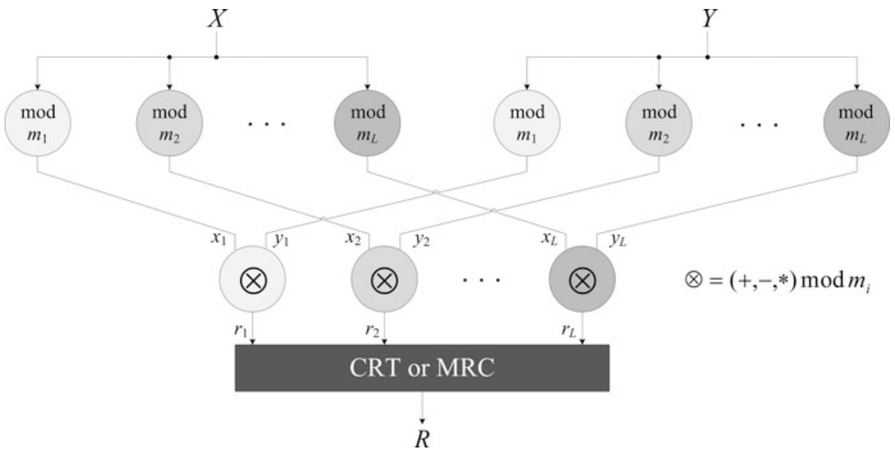


Fig. 2 General architecture of an RNS processor

modulus is r-bit long. To reconstruct an integer from its residues, two methods may be employed, namely the Chinese Remainder Theorem (CRT) and Mixed Radix Conversion (MRC) [57]. A general architecture of an RNS processor is depicted in Fig. 2.

3.2.1 The Chinese Remainder Theorem

CRT is based on the following equation

$$X = \left\langle \sum_{i=1}^{L} \left\langle x_i \cdot M_i^{-1} \right\rangle_{m_i} \cdot M_i \right\rangle_M , \tag{14}$$

where $M_i = M/m_i$ and M_i^{-1} is the inverse of M_i modulo m_i. An observation on (14) reveals the main characteristics and implications of CRT realization. CRT is a modulo multiply-accumulate procedure decomposed in:

- L inner modular products of small r-bit quantities to formulate $\langle x_i \cdot M_i^{-1} \rangle_{m_i}$
- L non-modular multiplications to formulate $\langle x_i \cdot M_i^{-1} \rangle_{m_i} \cdot M_i$
- addition of the previous results modulo M

It is apparent that CRT suffers from the large multiplications in the second step (note that M_is are $r(L-1)$-bit long) but also from the large modulo M addition in the final step. The addition of the inner products can be achieved in $O(\log L)$ time using standard addition techniques.

Important reductions of the complexity of this schema have been carried-out during the last 20 years [24, 37, 50, 51, 60, 61]. In addition to reducing the complexity of inner product calculation, they substitute the final modulo M addition with smaller and simpler operations.

The works in [24, 50] are based on the observation that the result in (14) is congruent modulo M. In order to obtain the exact value of X, we must compute

$$X = \sum_{i=1}^{L} \left\langle x_i \cdot M_i^{-1} \right\rangle_{m_i} \cdot M_i - \gamma M, \tag{15}$$

where γ is an integer correction factor. In other words, the large modulo M addition is replaced by a subtraction and a multiplication. Efficient calculation of the correction factor γ is critical. Shenoy and Kumaresan [50] developed an algorithm which requires a redundant modulus, $m_r \geq L$, so that the RNS base $\mathcal{M}$ is extended to $\mathcal{M} = (m_1, m_2, \ldots, m_L \mid\mid m_r)$. This adds an extra channel of calculations. Let X be an integer with an RNS representation $X_{\mathcal{M}} = (x_1, x_2, \ldots, x_L \mid\mid x_r)$, where $x_r = \langle X \rangle_{m_r}$. By reducing both sides of (15) mod m_r, we obtain

$$\begin{aligned}
\langle X \rangle_{m_r} &= \left\langle \left\langle \sum_{i=1}^{L} \left\langle x_i \cdot M_i^{-1} \right\rangle_{m_i} \cdot M_i \right\rangle_{m_r} - \langle \gamma M \rangle_{m_r} \right\rangle_{m_r} \Rightarrow \\
\langle \gamma \rangle_{m_r} &= \left\langle \left\langle M^{-1} \right\rangle_{m_r} \left(\left\langle \sum_{i=1}^{L} \left\langle x_i \cdot M_i^{-1} \right\rangle_{m_i} \cdot M_i \right\rangle_{m_r} - x_r \right) \right\rangle_{m_r} = \\
&= \left\langle \left\langle M^{-1} \right\rangle_{m_r} (\delta - x_r) \right\rangle_{m_r},
\end{aligned} \tag{16}$$

where $\delta = \left\langle \sum_{i=1}^{L} \left\langle x_i \cdot M_i^{-1} \right\rangle_{m_i} \cdot M_i \right\rangle_{m_r}$. Since $\gamma < L$ and $m_r \geq L$, it follows that $\gamma = \langle \gamma \rangle_{m_r}$ [50]. As all terms on the right hand side of (16) are known, the correction factor γ can be substituted in (15) to obtain X. Kawamura et al. [24] employed a different approach for the γ calculation. Starting again from (15) and using

$$\xi_i = \left\langle x_i \cdot M_i^{-1} \right\rangle_{m_i}, \tag{17}$$

we obtain

$$X = \sum_{i=1}^{L} \xi_i \cdot M_i - \gamma M. \tag{18}$$

Dividing both sides by M, we obtain

$$\sum_{i=1}^{L} \frac{\xi_i}{m_i} = \frac{X}{M} + \gamma. \tag{19}$$

Since $0 \leq X/M < 1$, it holds that $\gamma \leq \sum_{i=1}^{L} \frac{\xi_i}{m_i} < \gamma + 1$. Therefore,

$$\gamma = \left\lfloor \sum_{i=1}^{L} \frac{\xi_i}{m_i} \right\rfloor \tag{20}$$

with $0 \leq \gamma < L$, since $0 \leq \xi_i/m_i < 1$. Two approximations were employed to avoid hard divisions in (20). The denominator m_i is replaced by 2^r, where $2^{r-1} < m_i \leq 2^r$, while the numerator ξ_i is approximated by its most significant q bits, where $q < r$. Thus, instead of γ, an approximated value γ^* can be calculated by

$$\gamma^* = \left\lfloor \sum_{i=1}^{L} \frac{trunc(\xi_i)}{2^r} + \alpha \right\rfloor, \tag{21}$$

where $trunc(\xi_i) = \xi_i \wedge \overbrace{(1 \ldots 1)}^{q}\overbrace{(0 \ldots 0)}^{(r-q)}$ and $\wedge$ denotes an AND operation. An offset value $0 \leq \alpha < 1$ is introduced to compensate the error produced by the approximations. Since division by powers of 2 are simple shifts, (21) can be realized by additions alone. The offset value α can be determined so that the error issued by the approximations is zero [24].

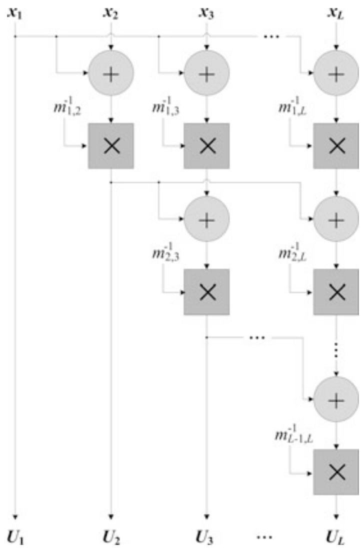


Fig. 3 The MRC process

3.2.2 Mixed-Radix Conversion

Another popular method for residue-to-decimal conversion is through the Mixed-Radix Conversion (MRC) algorithm (Fig. 3) [27]. The MRC of an integer X with an RNS representation $X_{\mathcal{M}} = (x_1, x_2, \ldots, x_L)$ is

$$X = U_1 + W_2 U_2 + \cdots + W_L U_L, \tag{22}$$

where $W_i = \prod_{j=2}^{i-1} m_j, \forall i \in [2, L]$ and $W_1 = 1$; The mixed-radix digits $U_1, U_2, \ldots, U_L$ are referred as the Mixed-Radix System (MRS) representation of X and can be computed as

$$\begin{aligned}
U_1 &= x_1 \\
U_2 &= \left\langle (x_2 - U_1)\, m_{1,2}^{-1} \right\rangle_{m_2} \\
U_3 &= \left\langle \left((x_3 - U_1)\, m_{1,3}^{-1} - U_2 \right) m_{2,3}^{-1} \right\rangle_{m_3} \\
&\ \vdots \\
U_L &= \left\langle \left(\ldots (x_L - U_1)\, m_{1,L}^{-1} - \cdots - U_{L-1} \right) m_{L-1,L}^{-1} \right\rangle_{m_L},
\end{aligned} \tag{23}$$

where $m_i m_{i,j}^{-1} \equiv 1 \mod m_j$. Equation (23) requires $L\frac{L-1}{2}$ modular multiplications. Another version of MRC that simplifies (23) and reduces the total number of modular multiplications to only $L-2$ is based on

$$\begin{aligned}
U_1 &= x_1 \\
U_2 &= \langle x_2 - x_1 \rangle_{m_2} \\
U_3 &= \langle x_3 - x_1 - W_2 U_2 \rangle_{m_3} \\
&\ \vdots \\
U_L &= \langle x_L - x_1 - W_2 U_2 - W_3 U_3 - \cdots - W_{L-1} U_{L-1} \rangle_{m_L},
\end{aligned} \tag{24}$$

provided that the predetermined factors $V_1 \equiv 1$ and $V_i \equiv \left\langle \left(\prod_{j=1}^{i-1} m_j \right)^{-1} \right\rangle_{m_i} = 1, \ \forall i \in [2, L]$ [63].

The main characteristic of MRC is its sequential nature. As any term U_i can't be calculated before U_{i-1} is available, the delay of the scheme is $O(L)$. In practical cryptosystem implementations, (15) is preferred since it avoids the large $\mod M$ reduction of (14) [4, 14, 15, 24]. MRC has also contributed to realistic cryptosystem implementations as in [48], where a matrix-based decomposition of operations was presented to simplify the conversion process.

3.3 Base Extension

Base extension (BE) is a critical operation when RNS is employed in cryptographic applications. Base extension increases the dynamic range of an RNS system by adding moduli to a predefined base. In real implementations, this is equivalent to extending an integer expressed in an RNS base to another RNS base representation. Let us assume two RNS bases $\mathcal{M} = (m_1, m_2, \ldots, m_L)$ and $\mathcal{M}\simeq = (m_{L+1}, m_{L+2}, \ldots, m_K)$ such as $\gcd(m_i, m_j) = 1, \forall(i,j) \in [1, K]$ and $K > L$. Assume also that the moduli are ordered from the smaller to the larger one, i.e., $m_1 < m_2 < \ldots m_L < m_{L+1} < m_{L+2} < \cdots < m_K$ (a restriction very easy to satisfy). Then, base extension of a number X expressed in base $\mathcal{M}$ can be defined as the transformation operation

$$X \equiv X_{\mathcal{M}} \xrightarrow{BE} X' \equiv X_{\mathcal{M}\simeq} | X \equiv X' \mod M \tag{25}$$

To accomplish such transformations, it is required that X is obtained from $X_{\mathcal{M}}$ and then $X_{\mathcal{M}\simeq}$ is calculated from X. In other words, a base extension can be seen as a residue-to-decimal operation followed by a decimal-to-residue operation in the new RNS system. From a complexity point of view, and compared to a single residue-to-binary operation, base extension requires additional efforts. Base extension techniques are reviewed in the following sections.

3.3.1 Szabo-Tanaka Method

The Szabo-Tanaka method for the BE operation is actually the MRC process expressed in (22) and (23) slightly modified to accommodate the calculation of an RNS digit $x_i, i \in [L+1, K]$ in the new RNS base $\mathcal{M}\simeq$. Let us use the RNS digit x_{L+1} as an example. The calculation is as follows

$$x_{L+1} = \left\langle \underbrace{U_1 + U_2 W_2 + \cdots + U_L W_L}_{MRC} \right\rangle_{m_{L+1}} \tag{26}$$

which can be rewritten as

$$\begin{aligned} x_{L+1} &= \Big\langle U_L W_L + \big\langle U_{L-1} W_{L-1} + \cdots + \langle U_3 W_3 + U_2 W_2 + U_1 \rangle_{m_{L+1}} \cdots \big\rangle_{m_{L+1}} \Big\rangle_{m_{L+1}} \\ &= \Big\langle U_L \langle W_L \rangle_{m_{L+1}} + \big\langle U_{L-1} \langle W_{L-1} \rangle_{m_{L+1}} + \cdots + \langle U_3 \langle W_3 \rangle_{m_{L+1}} + \\ &\quad + U_2 \langle W_2 + \rangle_{m_{L+1}} + U_1 \rangle_{m_{L+1}} \cdots \big\rangle_{m_{L+1}} \Big\rangle_{m_{L+1}} \end{aligned} \tag{27}$$

The advantage of (27) is that the calculation of x_{L+1} can be overlapped with the computation of the mixed radix coefficients, by using them as soon as they become

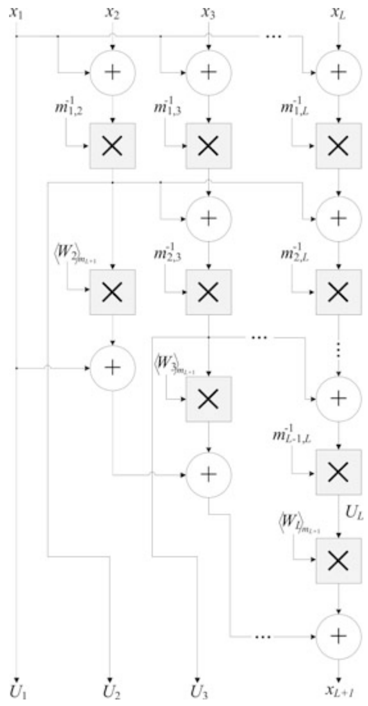


Fig. 4 Szabo-Tanaka base extension

available. Figure 4 shows this procedure. The pre-computed constants $\langle W_i \rangle_{m_{L+1}}$ are issued so that all operands participating in the base extension process have a word length equal to the modulus word length, resulting in more compact designs.

3.3.2 Redundant Modulus Method

We have already referred to the process proposed by Shenoy and Kumaresan [50] for residue-to-decimal conversion. In the same context, the authors presented a methodology for fast base extension, using an extra modulus. Starting from (15), we consider again the base extension to a modulus m_{L+1} of the new RNS base $\mathcal{M}\simeq$. The desired residue x_{L+1} is

$$x_{L+1} = \left\langle \left\langle \sum_{i=1}^{L} \left\langle x_i \cdot M_i^{-1} \right\rangle_{m_i} \cdot M_i \right\rangle_{m_{L+1}} - \langle \gamma M \rangle_{m_{L+1}} \right\rangle_{m_{L+1}} \tag{28}$$

The correction factor γ can be calculated from (16), since it depends only on the known residues x_i, $\forall i \in [1, L]$, and the redundant residue x_r. The value of γ can then

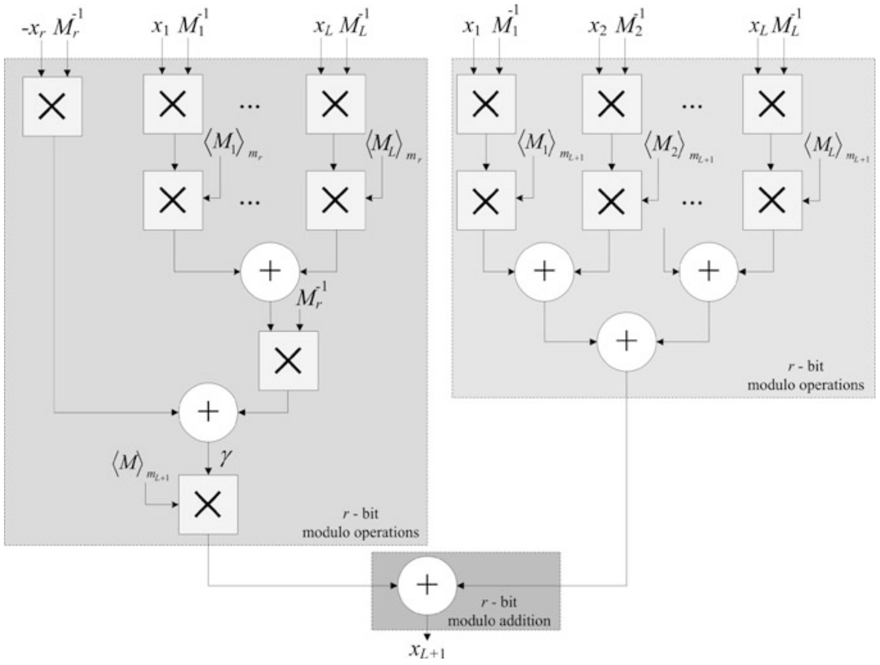

$-x_r\ M_r^{-1}$
$x_1\ M_1^{-1}$
$x_L\ M_L^{-1}$
$\langle M_1\rangle_{m_r}$
$\langle M_L\rangle_{m_r}$
M_r^{-1}
$\langle M\rangle_{m_{L+1}}$
γ
r - bit modulo operations
$x_1\ M_1^{-1}$
$x_2\ M_2^{-1}$
$x_L\ M_L^{-1}$
$\langle M_1\rangle_{m_{L+1}}$
$\langle M_2\rangle_{m_{L+1}}$
$\langle M_L\rangle_{m_{L+1}}$
r - bit modulo operations
r - bit modulo addition
x_{L+1}

Fig. 5 Shenoy-Kumaresan redundant modulus method for base extension

be substituted in (28) to calculate the residue x_{L+1}. The same process can be carried out simultaneously for the moduli $m_i, \forall i \in [L+1, K]$ of the new RNS base. A fully parallel architecture for the evaluation of x_{L+1} is illustrated in Fig. 5.

3.3.3 Kawamura et al. Base Extension

In 2000, Kawamura et al. presented a method for base extension that more or less defined the modern standards for a realistic RNS deployment in cryptographic hardware [24]. Their approximation method has been already presented in the conversions Sect. 3.2.1. The corresponding base extension algorithm is illustrated below as Algorithm 2 (we assume that the two bases have the same number of moduli L); the algorithm cleverly combines the calculation of the correction factor γ presented in (17)–(21) with the modulo reduction of the result by the moduli of the new RNS base.

The upper-bound of $\gamma < L$ in "Kawamura et al." method [24], is identical to the upper-bound obtained by the "Shenoy-Kumaresan" method [50] and much lower than the "Posch and Posch" method ($G < Lm_{max}$) [38, 39] (Fig. 6).

Algorithm 2 Base extension algorithm by Kawamura et al. [24]

```
Input: $X_{\mathcal{M}\simeq} = (x'_1, x'_2, \ldots, x'_L), \mathcal{M}, \mathcal{M}\simeq, \alpha$
Output: $X_{\mathcal{M}} = (x_1, x_2, \ldots, x_L)$
Precompute: $(M_i'^{-1})_{m'_i}, (M'_i)_{\mathcal{M}}$ $(\forall i = 1 \ldots L)$,$(-M')_{\mathcal{M}}$
 1: $\sigma_0 = \alpha$
 2: for all $i = 1 \ldots L$ do
 3:     $\xi_i = \langle x'_i \cdot M_i'^{-1} \rangle_{m'_i}$
 4:     $\delta_{i,0} = 0$
 5: end for
 6: for all $i = 1 \ldots L$ do
 7:     for $j = 1 \ldots L$ do
 8:         $\sigma_j = \sigma_{(j-1)} + \mathrm{trunc}(\xi_j)/2^r$
 9:         $\gamma_j^* = \lfloor \sigma_j \rfloor, \{\gamma_j^* = \{0,1\}\}$
10:         $\sigma_j = \sigma_j - \gamma_j^*$
11:         $\delta_{i,j} = \delta_{i,(j-1)} + \xi_j \cdot \langle M'_j \rangle_{m_i} + \gamma_j^* \cdot \langle -M' \rangle_{m_i}$
12:     end for
13: end for
14: for all $i = 1 \ldots L$ do
15:     $x_i = \langle \delta_{i,L} \rangle_{m_i}$
16: end for
```

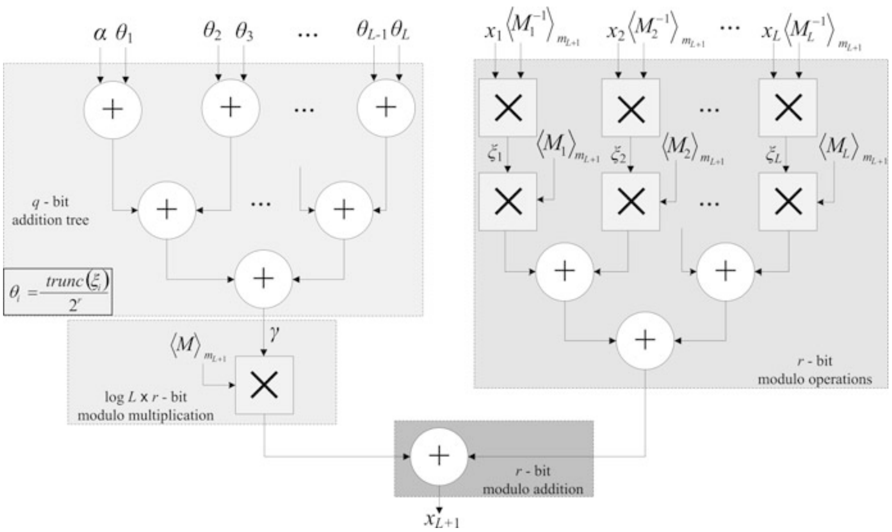


Fig. 6 "Kawamura et al." approximated base extension

3.4 Polynomial Residue Number System

Similar to RNS, a Polynomial RNS (PRNS) is defined through a set of L, pair-wise relatively prime polynomials $\mathcal{M} = (m_1(x), m_2(x), \ldots, m_L(x))$. We denote by

$M(x) = \prod_{i=1}^{L} m_i(x)$ the dynamic range of the PRNS. In PRNS, every polynomial $z(x) \in GF(2^n)$, with $\deg\{z(x)\} < \deg\{M(x)\}$, has a unique PRNS representation:

$$z_{\mathcal{M}} = (z_1, z_2, \ldots, z_L)\,, \tag{29}$$

such as $z_i = z(x) \bmod m_i(x)$, $i \in [1, L]$, denoted as $\langle z \rangle_{m_i}$. For simplicity, in the rest of this chapter, the notation "(x)" to denote polynomials shall be omitted. The notation z will be used interchangeably to denote either an integer z or a polynomial $z(x)$, according to context.

Assuming $a_{\mathcal{M}} = (a_1, a_2, \ldots, a_L)$ and $b_{\mathcal{M}} = (b_1, b_2, \ldots, b_L)$ as the PRNS representation of two polynomials $a, b \in GF(2^n)$, then all operations $\otimes \in (+, -, *)$ can be performed in parallel, as

$$a_{\mathcal{M}} \otimes b_{\mathcal{M}} = (\langle a_1 \otimes b_1 \rangle_{m_1}, \langle a_2 \otimes b_2 \rangle_{m_2}, \ldots, \langle a_L \otimes b_L \rangle_{m_L})\,. \tag{30}$$

Conversion from PRNS to weighted polynomial representation is identical to the MRC for integers. The only difference is that, the subtractions in (23) and (24) are substituted by polynomial additions. In the case of CRT for polynomials, the conversion is based on

$$z(x) = \sum_{i=1}^{L} \left\langle z_i(x) \cdot M_i^{-1}(x) \right\rangle_{m_i(x)} \cdot M_i(x), \tag{31}$$

where $M_i(x) = M(x)/m_i(x)$ and $M_i^{-1}(x)$ is the inverse of $M_i(x)$ modulo $m_i(x)$. Unlike the integer case in (14), the final reduction by the product polynomial $M(x)$ is not necessary in the case of polynomials over $GF(2^n)$.

4 RNS Modular Multiplication

Mathematician P.L. Montgomery presented in 1985 an ingenious method for efficient modular multiplication without trial divisions [35]. This 3-page paper is perhaps the most cited work in the field during the last 30 years and changed radically the available implementation options for efficient cryptosystem design. A modification of Montgomery's modular multiplication (MMM) algorithm to handle RNS data was proposed 10 years later by Karl C. Posch and Reinhard Posch [39]. Later, researchers produced more robust algorithms and implementations, mainly for use in the context of modular exponentiation for RSA [5, 14, 24]. Let us rewrite here the original MMM for convenience.

Algorithm 3 Montgomery modular multiplication [35]

```
Input: $a, b, N, R, R^{-1}$ { $a, b < N$ }
Output: $c \equiv abR^{-1} \mod N$, { $c < 2N$ }
 1: $s \leftarrow a \cdot b$
 2: $t \leftarrow s \cdot \left(-N^{-1}\right) \mod R$
 3: $u \leftarrow t \cdot N$
 4: $v \leftarrow s + u$
 5: $c \leftarrow v/R$
```

Condition $\gcd(R, N) = 1$ ensures the existence of $N^{-1} \mod R$. Condition $N < R$ is sufficient for $c < 2N$ since

$$c = \frac{ab + tN}{R} < \frac{N^2 + NR}{R} = \left(\frac{N}{R} + 1\right)N < 2N. \tag{32}$$

Since $cR = ab + tN$, $cR \equiv ab \mod N$ holds. By multiplying $R^{-1} \mod N$ on both sides of (32), $c \equiv abR^{-1} \mod N$, where R is the Montgomery radix.

The algorithm requires first to transform the input operands to their corresponding Montgomery representations [35]. Assuming an integer a, its Montgomery representation is defined as $\bar{a} = aR \mod N$. This conversion may be realized by means of an extra Montgomery multiplication by $R^2 \mod N$, i.e. $\bar{a} = a \times (R^2 \mod N) \times R^{-1} \mod p = aR \mod N$. With these inputs the algorithm outputs the Montgomery residue of the result, i.e., $\bar{c} = cR \mod N = abR \mod N$.

An extra Montgomery multiplication converts the Montgomery residue back to the integer domain representation. This iteration accepts as inputs the result $\bar{c} = cR \mod N$ of the Montgomery multiplication and $1 \mod N$ to produce $cR \times 1 \times R^{-1} \mod N = c \mod N$.

The challenges to transform this algorithm to RNS format are in steps 2 and 5. Step 2 is a modulo R operation. In non-RNS implementations, R is usually chosen to be a power of 2, thus modulo R operations amount to simple shifts. A method that provides the modulo operation of step 2 within RNS representation needs to be devised for the RNS version of the algorithm. Similar problems arise for step 5, where division by the Montgomery radix needs to be performed in RNS.

The trick to overcome these issues is to choose a new Montgomery radix. Assume that a base $\mathcal{M}\simeq = \{m'_1, m'_2, \ldots, m'_L\}$ is employed with a corresponding range M'. By assigning the Montgomery radix to be the range M' itself, step 2 is transformed to a step computed modulo M', since RNS is a closed modulo M' system. Unfortunately, computations in the same base cannot be continued, since in step 5 a quantity of the form $M'^{-1} \mod M'$ needs to be computed, which mathematically does not exist. For this reason, a new base $\mathcal{M} = \{m_1, m_2, \ldots, m_L\}$ is employed and a base extension from base $\mathcal{M}\simeq$ to base $\mathcal{M}$ is performed after step 2 of the algorithm.

Steps 3, 4, and 5 are computed in the new base $\mathcal{M}$, since now it is feasible to compute $M'^{-1} \mod M$ in step 5. The complete RNS Montgomery Modular

Multiplication (RNSMMM) algorithm is shown below as Algorithm 4. An extra BE is issued at the end of the algorithm so that inputs and outputs are compatible with each other, and the algorithm may be used repeatedly in the context of any modular exponentiation algorithm. This applies both to RSA and ECC applications, where consecutive modular multiplications constitute the heart of the cryptosystems' design. Steps performed in both bases are denoted as $\mathcal{T} = \mathcal{M} \cup \mathcal{M}\simeq$. Clearly, the complexity of the algorithm depends on the BE steps.

Algorithm 4 RNS Montgomery modular multiplication

```
Input: $a_{\mathcal{T}}, b_{\mathcal{T}}$ { $a, b < 2N$ }
Output: $c_{\mathcal{T}}$, { $c < 2N$ and $c \equiv abM'^{-1} \mod N$ }
Precompute: $\left(-N^{-1}\right)_{\mathcal{M}\simeq}, M'^{-1}_{\mathcal{M}}, N_{\mathcal{M}}$
1: $s_{\mathcal{T}} \leftarrow a_{\mathcal{T}} \cdot b_{\mathcal{T}}$
2: $t_{\mathcal{M}\simeq} \leftarrow s_{\mathcal{M}\simeq} \cdot \left(-N^{-1}\right)_{\mathcal{M}\simeq}$
3: $t_{\mathcal{M}} \leftarrow t_{\mathcal{M}\simeq}$ { base extension step }
4: $u_{\mathcal{M}} \leftarrow t_{\mathcal{M}} \cdot N_{\mathcal{M}}$
5: $v_{\mathcal{M}} \leftarrow s_{\mathcal{M}} + u_{\mathcal{M}}$
6: $c_{\mathcal{M}} \leftarrow v_{\mathcal{M}} \cdot M'^{-1}_{\mathcal{M}}$
7: $c_{\mathcal{M}\simeq} \leftarrow c_{\mathcal{M}}$ { base extension step}
```

We have already referred to the base extension options in Sect. 3.3. When it comes to cryptosystem design, three algorithms for RNSMMM have been proposed since the 2000s, which either employ these BE options or modified versions of them. Kawamura et al. [24] proposed an RNSMMM algorithm that utilizes their BE Algorithm 2, while the RNSMMM proposed by Bajard et al. [5] utilizes the "Shenoy-Kumaresan" BE method. Finally, Gandino et al. offered optimizations for both algorithms by reducing the total number of steps required using precomputations [14]. We analyze these options below.

4.1 RNS Montgomery Multiplication by Bajard et al.

Bajard et al. employed two different methods for the first and second base extension of the RNSMMM [5]. The first, shown below as Algorithm 5, involves an approximation error but performs faster than Kawamura's algorithm, while the second, depicted as Algorithm 6, is the one originally proposed by Shenoy and Kumaresan [50] and corrects the previous result. The two algorithms convert the corresponding quantities in Algorithm 4, i.e., the values of t and c respectively. Note also that the first algorithm is nothing more than the CRT expression in (14), while the second computes the correction factor γ of the CRT expression in (15).

The key idea is that if larger bases are selected, such as $M, M' > N(L + 2)^2$, the restriction to obtain a correct result during the first base extension can be relaxed without affecting the final result [5].

Algorithm 5 First BE algorithm by Bajard et al. [5]

```
Input: $t_{\mathcal{M}\simeq} = \left(t'_1, t'_2, \dots, t'_L\right)$
Output: $t_{\mathcal{M}\cup m_r} = (t_1, t_2, \dots, t_L, t_r)$
Precompute: $\left\langle {M'}_i^{-1} \right\rangle_{m'_i}$ $(\forall i = 1 \dots L$, $\left(M'_i\right)_{\mathcal{M}\cup m_r}$
1: $\xi_i = t'_i \cdot \left\langle {M'}_i^{-1} \right\rangle_{m'_i}, \forall i = 1 \dots L$
2: for all $j = 1 \dots L$ and $j = r$ do
3:     $t_j = \left\langle \sum_{i=1}^{L} \xi_i \cdot M'_i \right\rangle_{m_j}$
4: end for
```

Algorithm 6 Second BE algorithm by Bajard et al. [5]

```
Input: $c_{\mathcal{M}\cup m_r} = (c_1, c_2, \dots, c_L, c_r)$
Output: $c_{\mathcal{M}\simeq} = \left(c'_1, c'_2, \dots, c'_L\right)$
Precompute: $\left\langle M_j^{-1} \right\rangle_{m_j}$ $(\forall j = 1 \dots L, r)$,$(-M)_{\mathcal{M}\simeq}$, $\left(M_j\right)_{\mathcal{M}\simeq\cup m_r}$ $(\forall j = 1 \dots L)$
1: for all $j = 1 \dots L$ do
2:     $\tilde{c}_j = \left\langle c_j \cdot M_j^{-1} \right\rangle_{m_j}$
3: end for
4: $c''_r = \left\langle \sum_{j=1}^{L} \tilde{c}_j \cdot M_j \right\rangle_{m_r}$
5: $\gamma = \langle (c''_r - c_r) M_r^{-1} \rangle_{m_r}$
6: for all $i = 1 \dots L$ do
7:     $c'_i = \left\langle \sum_{j=1}^{L} \tilde{c}_j \cdot M_j \right\rangle_{m'_i}$
8: end for
9: for all $i = 1 \dots L$ do
10:    $c'_i = \langle c'_i - \gamma M \rangle_{m'_i}$
11: end for
```

4.2 RNS Montgomery Multiplication by Gandino et al.

Gandino et al. [14] improved both of the solutions presented previously by issuing pre-computation steps. In the following, values with a $\widehat{hat}$ symbol denote values multiplied by M_j^{-1} in base $\mathcal{M}$, where $M_j = M/m_j, \forall j = 1 \dots L$. Also, the form A_B^{-1} denotes the multiplicative inverse of A on B. The re-organized versions of Kawamura's et al. BE are shown below as Algorithms 7 and 8, while the re-organized versions of Bajard et al. BE correspond to Algorithms 9 and 10 respectively.

Interestingly, while the BE algorithms proposed by Kawamura et al. and Bajard et. al convert only one value at steps 3 and 7 of Algorithm 4, the corresponding reorganized versions proposed by Gandino et al. include in the BE algorithm all other steps of Algorithm 4 as well. This results in fewer modular multiplications required for one RNSMMM, as shown in Table 1 [15].

Algorithm 7 Reorganized first BE algorithm for Kawamura et al. [14]

```
**Input:** $s_{\mathcal{M}\simeq}, \alpha = 0, \hat{\hat{s}}_{\mathcal{M}}$
**Output:** $\hat{c}_{\mathcal{M}}$
**Precompute:** $\left(M'^{-1}_{M} M_j\right)_{\mathcal{M}}, \left(-NM_j\right)_{\mathcal{M}}, \left(M'_i N M'^{-1}_{M} M_j^{-1}\right)_{\mathcal{M}}, \left\langle -N^{-1} M'^{-1}_{i}\right\rangle_{m'_i} (\forall i = 1 \ldots L)$
 1: $\xi_i = \left\langle s_i \cdot \left(-N^{-1} M'^{-1}_i\right)\right\rangle_{m'_i}, \forall i = 1 \ldots L$
 2: $\sigma = \alpha$
 3: $\hat{c}_j = \left\langle \hat{\hat{s}}_j \cdot \left(M'^{-1}_{M} M_j\right)\right\rangle_{m_j}, \forall j = 1 \ldots L$
 4: for $i = 1 \ldots L$ do
 5:     $\sigma = \sigma + \text{trunc}(\xi_i)/2^r$
 6:     $\gamma^* = \lfloor \sigma \rfloor$
 7:     $\sigma = \sigma - \gamma^*$
 8:     $\hat{c}_j = \left\langle \hat{c}_j + \xi_i \cdot \left(M'_i N M'^{-1}_{M} M_j^{-1}\right) + \gamma^* \cdot \left(-N M_j^{-1}\right)\right\rangle_{m_j}, \forall j = 1 \ldots L$
 9: end for
```

Algorithm 8 Reorganized second BE algorithm for Kawamura et al. [14]

```
**Input:** $\hat{c}_{\mathcal{M}}, \alpha = 0.5$
**Output:** $c_{\mathcal{M}\simeq}$
**Precompute:** $\langle M_j\rangle_{\mathcal{M}\simeq}, (-M)_{\mathcal{M}\simeq}$
 1: $\sigma = \alpha$
 2: $c_i = 0, \forall i = 1 \ldots L$ in $\mathcal{M}\simeq$
 3: for $j = 1 \ldots L$ do
 4:     $\sigma = \sigma + \text{trunc}(\hat{c}_j)/2^r$
 5:     $\gamma^* = \lfloor \sigma \rfloor$
 6:     $\sigma = \sigma - \gamma^*$
 7:     $c_i = \langle c_i + \hat{c}_j \cdot M_j + \gamma^* \cdot (-M)\rangle_{m'_i}, \forall i = 1 \ldots L$
 8: end for
```

Algorithm 9 Reorganized first BE algorithm for Bajard et al. [5]

```
**Input:** $s_{\mathcal{M}\simeq\cup m_r} = (s_1, s_2, \ldots, s_L, s_r), \hat{\hat{s}}_{\mathcal{M}}$
**Output:** $\hat{c}_{\mathcal{M}}, c_r$
**Precompute:** $\left(M'^{-1}_{M} M_j\right)_{\mathcal{M}}, \left\langle M'^{-1}_{M}\right\rangle_{m_r}, \left\langle M'_i N M'^{-1}_{M}\right\rangle_{m_r} (\forall i = 1 \ldots L), \left(M'_i N M'^{-1}_{M} M_j^{-1}\right)_{\mathcal{M}}$
    $(\forall i = 1 \ldots L), \left\langle -N^{-1} M'^{-1}_{i}\right\rangle_{m'_i} (\forall i = 1 \ldots L)$
 1: $m'_i = \left\langle s_i \cdot \left(-N^{-1} M'^{-1}_{i}\right)\right\rangle_{m'_i}, \forall i = 1 \ldots L$
 2: $\hat{c}_j = \left\langle \hat{\hat{s}}_j \cdot M'^{-1}_{M} M_j + \sum_{i=1}^{L} m'_i \cdot \left(M'_i N M'^{-1}_{M} M_j^{-1}\right)\right\rangle_{m_j}, \forall j = 1 \ldots L$
 3: $c_r = \left\langle s_r \cdot M'^{-1}_{M} + \sum_{i=1}^{L} m'_i \cdot \left(M'_i N M'^{-1}_{M}\right)\right\rangle_{m_r}$
```

Algorithm 10 Reorganized second BE algorithm for Bajard et al. [5]

```
Input: $\hat{c}_{\mathcal{M}} = (c_1, c_2, \ldots, c_L), \langle c \rangle_{m_r}$
Output: $c_{\mathcal{M}\simeq} = (c'_1, c'_2, \ldots, c'_L)$
Precompute: $\langle M_r^{-1} \rangle_{m_r}, (M_j)_{\mathcal{M}\simeq}\ (\forall j = 1 \ldots L), \langle M_j \rangle_{m_r}\ (\forall j = 1 \ldots L), (-M)_{\mathcal{M}\simeq}$
1: $c''_r = \left\langle \sum_{j=1}^{L} \hat{c}_j \cdot M_j \right\rangle_{m_r}$
2: $\gamma = \langle (c''_r - c_r) M_r^{-1} \rangle_{m_r}$
3: $c'_i = \left\langle \sum_{j=1}^{L} \hat{c}_j \cdot M_j \right\rangle_{m'_i}, \forall i = 1 \ldots L$
4: $c'_i = \langle c'_i - \gamma M \rangle_{m'_i}, \forall i = 1 \ldots L$
```

Table 1 Number of modular multiplications in state-of-the-art RNSMMM

Steps in RNSMMM	[24]	[5]	[14] applied in [24]	[14] applied in [5]
1, 3, 4	$5L$	$5L$	$2L$	$2L$
First BE	$L^2 + 2L$	$L^2 + L$	$L^2 + 3L$	$L^2 + 2L$
Second BE	$L^2 + 2L$	$L^2 + 2L$	$L^2 + L$	$L^2 + L$
Total	$2L^2 + 9L$	$2L^2 + 8L$	$2L^2 + 6L$	$2L^2 + 5L$

4.3 Modular Reduction by the RNS Moduli

The modular reduction technique by each RNS modulus is identical for all the works in [5, 14, 24], since not only it offers simple implementations but also allows for fair comparisons. Assuming moduli of the form $m_i = 2^r - c_i$, where $c_i < 2^h$ and $h < \frac{r-1}{2}$, the reduction of an integer $X < 2^{2r}$ requires two multiplications and three additions according to

$$y = X \mod 2^r + ((X << r) \mod 2^r) \cdot c_i + (X << 2r) \cdot c_i^2, \tag{33}$$

where $<<$ denotes a left-shift operation [14].

4.4 Conversions to/from RNS

Kawamura et al. offered also an efficient method for conversions to/from RNS representations [24]. The key-idea was to employ high-radix representations so that each high-radix digit can be assigned to an RNS channel. In that way handling of very large integers can be achieved. A radix-2^r representation of an integer X as a L-tuple $\left(x^{(L-1)}, \ldots, x^{(0)}\right)$ satisfies

$$X = \sum_{i=0}^{L-1} x^{(i)} 2^{ri} = \left(2^{r(L-1)}, \ldots, 2^r, 1\right) \begin{bmatrix} x^{(L-1)} \\ \vdots \\ x^{(1)} \\ x^{(0)} \end{bmatrix}, \tag{34}$$

where $0 \le x^{(i)} \le 2^r - 1$. By applying the modulo m_j operation in (34) we can convert the integer X to its associated RNS representation by

$$\langle X \rangle_{m_j} = \left\langle \sum_{i=0}^{L-1} x^{(i)} \left\langle 2^{ri} \right\rangle_{m_j} \right\rangle_{m_j}, \forall j \in [1, L]. \tag{35}$$

If constants $\left\langle 2^{ri} \right\rangle_{m_j}$ are precomputed, this computation is a typical multiply-accumulate operation and can be computed in L steps, when executed by L units in parallel.

As (15) is the basis of the presented RNSMMM algorithms, it is useful to employ it also for the RNS-to-decimal conversion. Let us rewrite (15) as

$$\begin{aligned} X &= \sum_{i=1}^{L} \left\langle x_i \cdot M_i^{-1} \right\rangle_{m_i} \cdot M_i - \gamma M = \\ &= \left(2^{r(L-1)}, \ldots, 2^r, 1\right) \sum_{i=1}^{L} \left\{ \xi_i \cdot \begin{bmatrix} M_{i(L-1)} \\ \vdots \\ M_{i(1)} \\ M_{i(0)} \end{bmatrix} - \gamma \begin{bmatrix} M_{(L-1)} \\ \vdots \\ M_{(1)} \\ M_{(0)} \end{bmatrix} \right\}, \end{aligned} \tag{36}$$

where $\xi_i = \left\langle x_i \cdot M_i^{-1} \right\rangle_{m_i}$. As soon as γ has been evaluated using the methods of section 3, each row of (36) can be computed in parallel in each modulus channel by means of multiply-accumulate operations. In this case, carry should be propagated from channel 1 until channel L [24].

All works in [5, 14, 24] utilize cell-based architectures for implementing the algorithms in [24] and [5, 14] respectively. Each cell corresponds to a single RNS modulus and utilizes a multiply-accumulate (MAC) unit followed by a modular reduction unit which performs reduction by the corresponding RNS modulus using (33). Actually, with slight modifications, the architecture in [14] supports both algorithms in [5, 24]. The cell structure is shown in Fig. 7 [14]; a common bus that connects the cells as well as lines connecting one cell to a subsequent one are omitted, for simplicity reasons. The multiply-accumulate unit is depicted at the top of the cell and the modular reduction units at the bottom are a straightforward imple-

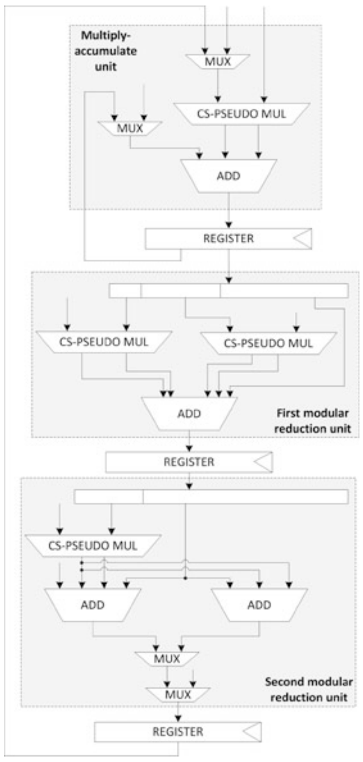


Fig. 7 MAC cell [15]

mentation of (33). An in-depth analysis on the number of clock-cycles required to accomplish all operations along with an architectural comparison presentation can be found in [14].

5 Dual-Field Residue Arithmetic Architectures

CRT has dominated the available options for base extension and residue-to-binary conversion since the introduction of RNS in cryptography. However, MRC has been recently employed as it avoids the evaluation of the γ factor of (15) and can form the basis of alternative RNS-based Montgomery multiplication [44, 48]. The derived algorithm is identical to Algorithm 4, however the BE algorithm is now based on the modified version of MRC shown in (22) and (24). Compared to the "Szabo-Tanaka" approach in (23), which requires $L\frac{L-1}{2}$modular multiplications, the optimized MRC requires only $L-2$ modular multiplications.

Algorithm 11 depicts the base extension process that converts an integer X expressed in an RNS base $\mathcal{M}\simeq$ as $X_{\mathcal{M}\simeq}$ to the RNS representation of another base $\mathcal{M}$. It implements (24) in steps 1–8 to obtain the mixed-radix digits U_i of X. In steps 9–15, (22) is realized, while the whole summation is computed modulo each modulus m_i of the new base $\mathcal{M}$.

Algorithm 11 RNSMMM base extension based on MRC

```
Input: $X_{\mathcal{M}\simeq} = \left(x'_1, x'_2, \cdots, x'_L\right), \mathcal{M}, \mathcal{M}\simeq$
Output: $X_{\mathcal{M}} = (x_1, x_2, \cdots, x_L)$
 1: $W_1 \leftarrow 0$
 2: $U_1 \leftarrow x'_1$
 3: for all $i = 2, \ldots, L$ do
 4:     $U_i \leftarrow x'_i - x'_1$
 5:     for $j = 1$ to $i - 1$ do
 6:         $U_i \leftarrow \langle U_i - W_j U_j \rangle_{m'_i}$
 7:     end for
 8: end for
 9: $x_1 \leftarrow 0$
10: for all $i = 1, \ldots, L$ do
11:     for $j = 1$ to $L$ do
12:         $\kappa_{ij} = \langle W_j M_j \rangle_{m_i}$
13:         $x_i \leftarrow \langle \kappa_{ij} + x_i \rangle_{m_i}$
14:     end for
15: end for
```

Contrasted to the other algorithms that employ CRT and utilize approximation methods to compute the correction factor γ of (15) [5, 14, 15, 24], the MRC-based base extension is error-free. Conditions $\gcd(M', N) = 1$ and $\gcd(M, M') = 1$ are sufficient for the existence of $(-N^{-1})_{\mathcal{M}\simeq}$ and $M'^{-1}_{\mathcal{M}}$, respectively. As it holds that

$$c = \frac{v}{M'} = \frac{ab + tN}{M'} < \frac{(2N)^2 + M'N}{M'} = \left(\frac{4N}{M'} + 1\right) N \leq 2N, \tag{37}$$

it follows that $4N \leq M'$ is sufficient for $c < 2N$ to hold when $a, b < 2N$. Finally, (37) also shows that $2N \leq M$ is sufficient for $c < M'$ and $v < MM'$. Since v is the maximum intermediate value, all values are less than MM' [24, 46].

5.1 Embedding PRNS in $GF(2^n)$ Montgomery Multiplication

Modifications of the Montgomery algorithm for multiplication in $GF(2^n)$ that encompass PRNS have also been proposed in the literature [6, 44, 48]. For example, the algorithm in [48] employs general polynomials (of any degree), and is an extension of an algorithm [6], which employs trinomials for the PRNS modulus set.

The PRNSMMM algorithm for $GF(2^n)$ is illustrated as Algorithm 12, while the corresponding algorithm for base extension in $GF(2^n)$ is identical to Algorithm 11.

Algorithm 12 Polynomial RNSMMM in $GF(2^n)$

```
Input: $a_{\mathcal{T}}, b_{\mathcal{T}}, N^{-1}_{\mathcal{M}\simeq}, M'^{-1}_{\mathcal{M}}, N_{\mathcal{M}}$, $\{\deg\{a\} < n, \deg\{b\} < n\}$
Output: $c_{\mathcal{T}}$, $\{\deg\{c\} < n, c = abM'^{-1} \mod N\}$
 1: $s_{\mathcal{T}} \leftarrow a_{\mathcal{T}} \cdot b_{\mathcal{T}}$
 2: $t_{\mathcal{M}\simeq} \leftarrow s_{\mathcal{M}\simeq} \cdot N^{-1}_{\mathcal{M}\simeq}$
 3: $t_{\mathcal{M}} \leftarrow t_{\mathcal{M}\simeq}$
 4: $u_{\mathcal{M}} \leftarrow t_{\mathcal{M}} \cdot N_{\mathcal{M}}$
 5: $v_{\mathcal{M}} \leftarrow s_{\mathcal{M}} + u_{\mathcal{M}}$
 6: $c_{\mathcal{M}} \leftarrow v_{\mathcal{M}} \cdot M'^{-1}_{\mathcal{M}}$
 7: $c_{\mathcal{M}\simeq} \leftarrow c_{\mathcal{M}}$
```

The only difference is that integer additions/subtractions and multiplications are replaced by polynomial ones. Again, the degrees of the input and output polynomials are both less than n, which allows the construction of a modular exponentiation algorithm by repetition of the PRNSMMM. Base extension in step 7 is employed for the same reason.

5.2 *Versatile Architectures*

An examination of the RNSMMM and PRNSMMM algorithms reveals potential for unification into a common Dual-field Residue Arithmetic MMM (DRAMMM) algorithm and a common Dual-field Base Extension (DBE) algorithm [44, 48]. The unified DRAMMM algorithm is depicted below as Algorithm 13, where $\oplus$ represents a dual-field addition/subtraction and $\odot$ represents a dual-field multiplication. For simplicity, the DBE algorithm is omitted since it is identical to the BE Algorithm 11, considering $\oplus$, $\odot$ dual-field operators instead of the typical additions/multiplications.

Both the DRAMMM and the DBE algorithm maintain the MAC characteristics of the previously presented BE algorithms and each channel handles operations of word length equal to the modulus word length r. This allows for a fully-parallel hardware implementation, employing parallel MAC units, each dedicated to a modulus of the RNS/PRNS base.

Algorithm 13 The dual-field residue arithmetic MMM algorithm

```
Input: $a_{\mathcal{T}}, b_{\mathcal{T}}, \left(-N^{-1}\right)_{\mathcal{M}\simeq}, M'^{-1}_{\mathcal{M}}, N_{\mathcal{M}}$, $\{\ a, b < 2N\ \}$
Output: $c_{\mathcal{T}}$, $\{\ c < 2N$ and $c \equiv abM'^{-1} \mod N\ \}$
 1: $s_{\mathcal{T}} \leftarrow a_{\mathcal{T}} \odot b_{\mathcal{T}}$
 2: $t_{\mathcal{M}\simeq} \leftarrow s_{\mathcal{M}\simeq} \odot \left(-N^{-1}\right)_{\mathcal{M}\simeq}$
 3: $t_{\mathcal{M}} \leftarrow t_{\mathcal{M}\simeq}$ { base extension step }
 4: $u_{\mathcal{M}} \leftarrow t_{\mathcal{M}} \odot N_{\mathcal{M}}$
 5: $v_{\mathcal{M}} \leftarrow s_{\mathcal{M}} \oplus u_{\mathcal{M}}$
 6: $c_{\mathcal{M}} \leftarrow v_{\mathcal{M}} \odot M'^{-1}_{\mathcal{M}}$
 7: $c_{\mathcal{M}\simeq} \leftarrow c_{\mathcal{M}}$ { base extension step}
```

It has been proved that a valid RNS/PRNS transformation of the Montgomery algorithm requires

$$\left.\begin{array}{rl} \deg\{M\} & > n \\ \deg\{M'\} & > n \\ M & \geq 2N \\ M' & \geq 4N \end{array}\right\}, \tag{38}$$

which means that RNS/PRNS ranges should be selected to be of word length

$$\begin{array}{l} \lceil \log M \rceil \geq \max\{\lceil \log 2N \rceil, n\} \\ \lceil \log M' \rceil \geq \max\{\lceil \log 4N \rceil, n\} \end{array} \tag{39}$$

for the bases $\mathcal{M}$ and $\mathcal{M}\simeq$, respectively [48]. DRAMMM and DBE algorithms along with conditions (38) and (39) provide the necessary conditions for a dual-field residue arithmetic Montgomery multiplication.

As described before, the structure of the DRAMMM algorithm allows it to be reused in the context of any exponentiation algorithm. A possible implementation is depicted in Algorithm 14, requiring in total $2n + 2$ DRAMMM multiplications [30, 32]. Using Fermat's little theorem, field inversion can be realized by field exponentiation [18]. Thus, it can be efficiently mapped to the DRAMMM architecture as well without extra hardware.

Algorithm 14 Modular exponentiation based on DRAMMM algorithm

```
Input: $z_T, e = (e_{n-1} \dots e_1 e_0)_2$
Output: $b_T, b \equiv \langle z^e \rangle_N$
1: $b \leftarrow 1$
2: for $i = n - 1, \dots, 0$ do
3:     $b \leftarrow DRAMMM(b, b)$
4:     if $e_i = 1$ then
5:         $b \leftarrow DRAMMM(b, z)$
6:     end if
7: end for
8: return $b$
```

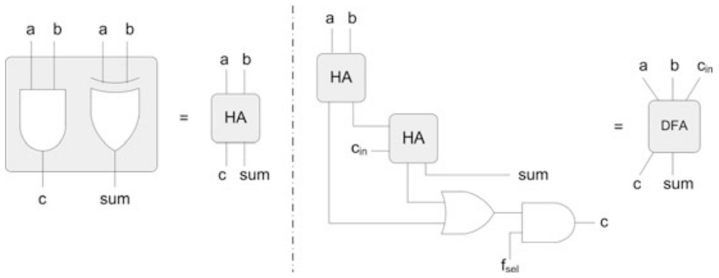


Fig. 8 Dual-field full-adder cell

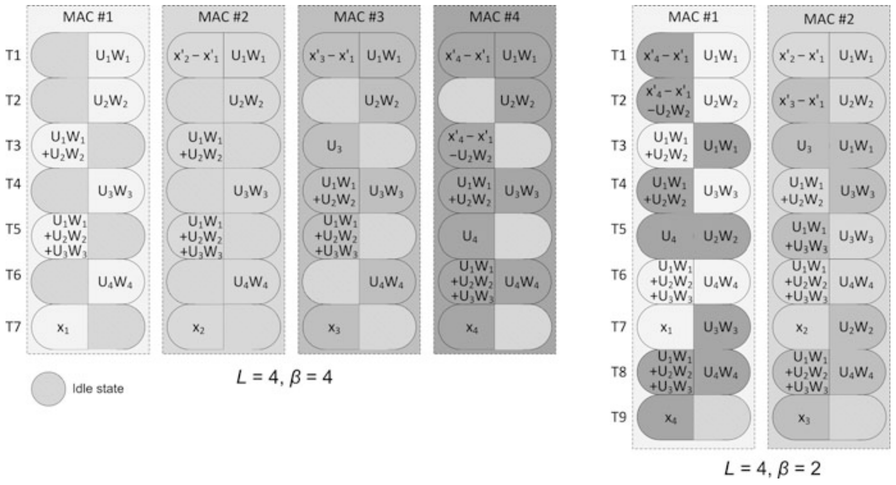


Fig. 9 Task distribution in DRAMMM algorithm [48]

5.2.1 Residue-to-Binary Conversion

As all operands in (24) are of word length r, they can be efficiently handled by an r-bit MAC unit. However, (22) employs multiplications with large values, namely the W_is. To overcome this problem, (22) can be rewritten in matrix notation, as in (40), which implies a fully parallel implementation of the conversion process [48].

The inner products of a row i are calculated in parallel in each MAC unit. Each MAC then propagates its result to the next MAC, so that at the end the last MAC(L) outputs the radix-2^r digit $z^{(i)}$ of the result. In parallel with this summation, inner products of the next row $i+1$ can be formulated, since the adder and multiplier of the DRAMMM MAC unit may operate in parallel.

$$
\begin{aligned}
z &= \left\{ U_1 \odot \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix} \oplus U_2 \odot \begin{bmatrix} 0 \\ \vdots \\ 0 \\ W_2^{(1)} \\ W_2^{(0)} \end{bmatrix} \oplus U_3 \odot \begin{bmatrix} 0 \\ \vdots \\ W_3^{(2)} \\ W_3^{(1)} \\ W_3^{(0)} \end{bmatrix} \oplus \cdots \oplus U_L \odot \begin{bmatrix} W_L^{(L-1)} \\ \vdots \\ W_L^{(2)} \\ W_L^{(1)} \\ W_L^{(0)} \end{bmatrix} \right\} \odot V \\
&= \begin{bmatrix} 0 \oplus & 0 & \oplus & 0 & \oplus \cdots \oplus U_L \odot W_L^{(L-1)} \\ \vdots \oplus & \vdots & \oplus & \vdots & \oplus \vdots \oplus \quad \vdots \\ 0 \oplus & 0 & \oplus & U_3 \odot W_3^{(2)} & \oplus \cdots \oplus U_L \odot W_L^{(2)} \\ 0 \oplus & U_2 \odot W_2^{(1)} & \oplus & U_3 \odot W_3^{(1)} & \oplus \cdots \oplus U_L \odot W_L^{(1)} \\ U_1 \oplus & U_2 \odot W_2^{(0)} & \oplus & U_3 \odot W_3^{(0)} & \oplus \cdots \oplus U_L \odot W_L^{(0)} \end{bmatrix} \odot V = \begin{bmatrix} z^{(L-1)} \\ \vdots \\ z^{(2)} \\ z^{(1)} \\ z^{(0)} \end{bmatrix} \odot V
\end{aligned}
\tag{40}
$$

5.3 Dual-Field Addition/Subtraction

A dual-field full-adder (DFA) cell is basically a full-adder cell equipped with a field-select signal (*fsel*) that controls the operation mode, as depicted in Fig. 8 [43]. When $fsel = 0$, the carry output is forced to 0 and the sum outputs the *XOR* operation of the inputs. As already mentioned, this is equivalent to the addition operation in $GF(2^n)$. When $fsel = 1$, $GF(p)$ mode is selected and the cell operates as a normal FA cell. Obviously, dual-field adders in various configurations (carry-propagate, carry-skip, etc) can be mechanized by utilizing DFA cells. With this cell serving as the basic constructing component, hardware modules for dual-field addition/subtraction (DMAS), dual-field multiplication (DM), and dual-field modular reduction can be mechanized [48].

5.4 Dual-Field Modular Reduction

A final modular reduction by each RNS/PRNS modulus is required, for each multiplication outcome, within each MAC unit. Assume a $2r$-bit product c that needs to be reduced modulo an integer modulus m_i. By selecting m_i of the form $2^r - \mu_i$, where the h-bit $\mu_i \ll 2^r$, the modular reduction process can be simplified as

$$\langle c\rangle_{m_i} = \left\langle \overbrace{\sum_{i=0}^{r-1} c_i 2^i}^{E} + 2^r \overbrace{\sum_{i=0}^{r-1} c_{r+i} 2^i}^{F} \right\rangle_{m_i} = \overbrace{\langle E + 2^r F\rangle_{m_i}}^{d} = \left\langle \overbrace{\sum_{i=0}^{r-1} d_i 2^i + \mu_i \sum_{i=0}^{h} d_{r+i} 2^i}^{\xi} \right\rangle_{m_i}. \tag{41}$$

From (41), it is apparent that

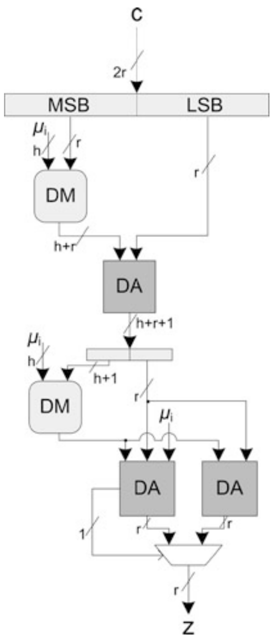
$$\langle c\rangle_{m_i} = \begin{cases} \sum_{i=0}^{r-1} \xi_i 2^i, \xi < 2^r - \mu \\ \sum_{i=0}^{r} \xi_i 2^i + \mu_i, 2^r - \mu < \xi < 2^r. \end{cases} \tag{42}$$

The same decomposition can be applied to polynomials and, consequently, if dual-field adders and dual-field multipliers are employed, a dual-field modular reduction (DMR) unit can be mechanized as shown in Fig. 10. The word length h of μ_i can be limited to a maximum of 10 bits for a base with 66 elements [24].

5.5 Multiply-Accumulate Unit

The circuit organization of the basic unit that performs all operations, the multiply-accumulate (MAC) unit, is shown in Fig. 11. Its operation is analyzed below in three steps, corresponding to the three phases of the calculations it handles, i.e., binary-to-residue conversion, RNS/PRNS Montgomery multiplication, and residue-to-binary conversion.

Fig. 10 Dual-field modular reduction unit (DMR)



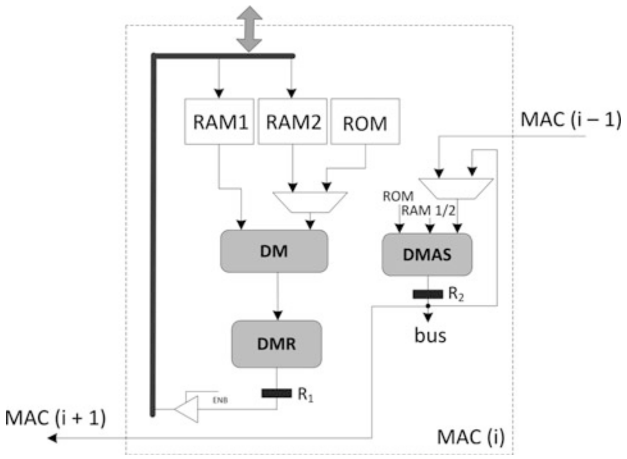


Fig. 11 The DRAMMM MAC unit

5.5.1 Binary-to-Residue Conversion

Initially, r-bit words of the input operands, as implied by (34), are cascaded to each MAC unit and stored in RAM1 at the top of Fig. 11. These words serve as the first input to the multiplier, along with the quantities $\langle 2^{ri}\rangle_{m_i,m_i'}$ or $\langle x^{ri}\rangle_{m_i,m_i'}$, which are stored in a ROM. Their multiplication produces the inner products of (34) which are added recursively in the DMAS unit. The result is stored via the bus in RAM1. The process is repeated for the second operand and the result is stored in RAM2, so that when the conversion is finished, each MAC unit contains the residue digits of the two operands in the two RAM units. The conversion requires L steps.

5.5.2 Montgomery Multiplication

The first step of DRAMMM is a modular multiplication of the residue digits of the operands. Since these digits are immediately available by the two RAMs, a modular multiplication is executed and the result in R_1 is stored in RAM1 for base $\mathcal{M}\simeq$ and in RAM2 for base $\mathcal{M}$. Step 2 of DRAMMM is a multiplication of the previous result with a constant provided by the ROM. The results correspond to x'_i inputs of the DBE algorithm and are stored again in RAM1. All MAC units are updated through the bus with the corresponding RNS digits of all other MACs and a DBE process is initiated.

To illustrate the DBE process, a task distribution graph is presented in Fig. 9 for a DRAMMM requiring $L = 4$ moduli. Two cases are represented; the first corresponds to a fully parallel architecture with $\beta = 4$ units and the second shows how the tasks can be overlapped when only $\beta = 2$ MAC units are available. Each MAC unit has been assigned to a different shading pattern, thus in the overlapped case the color codes signify when a MAC unit performs operations for other units. In the example of Fig. 9, MAC(1) handles MAC(4) and MAC(2) handles MAC(3).

In each cycle, modular additions and multiplications are performed in parallel in each MAC. To depict this, each cycle is split in two parts: the operations on the left correspond to modular additions and on the right to modular multiplications. The results obtained by each operation are depicted in each cycle (they correspond to Algorithm 11), while idle states are denoted by dashed lines. An analysis on the number of clock cycles required, and how MAC units can be efficiently paired is presented in the next section.

The remaining multiplications, additions, and the final base extension operation required by the DRAMMM algorithm are computed in the same multiply-accumulate manner and the final residue Montgomery product can be either driven to the I/O interface, or it can be reused by the MAC units to convert the result to binary format.

5.5.3 Residue-to-Binary Conversion

Residue-to-binary conversion is essentially a repetition of the DBE algorithm, except for steps 9–14, which are no longer modulo operations. Instead, (40) has been developed to map efficiently the conversion process to the DRAMMM architecture. An important observation is that, whenever the preceding operation of a residue-to-binary conversion is a DRAMMM, which ends with a DBE execution, time savings

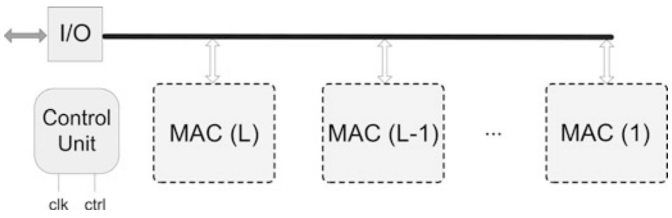


Fig. 12 DRAMMM general architecture

are achieved, since the upper part of the DBE algorithm (steps 1–8) is common with the conversion. Thus, the intermediate results from steps 1–8 can be stored until DBE is finished and then reused to implement (40).

To illustrate the conversion process assume the generation of the inner products in row 1 of (40). Each product is calculated in parallel in each MAC unit and a "*carry-propagation*" from MAC(1) to MAC(L) is performed to add all inner products. When summation finishes, the first digit $z^{(0)}$ of the result is produced in MAC(L). In parallel with this "*carry-propagation*", the inner products of line 2 are calculated. As soon as a MAC unit completes an addition of carry-propagated inner products for line 1, a new addition for line 2 is performed. The process continues for all lines of (40) and the result is available after L steps. The complete DRAMMM architecture is depicted in Fig. 12.

5.5.4 Delay

Assume that $\beta \leq L$ parallel MAC units are utilized in a real implementation. To simplify the discussion, it is assumed that β is a multiple of L, i.e., $L = k\beta$. This means that each one of the first MAC(i), $i = 1, 2, \ldots, \beta$, will provide results for $k-1$ more channels. By construction of the MRC, the DBE process in Fig. 9 requires $2L - 1$ clock cycles in the full parallel case. Each channel $1 \leq i \leq \beta$ requires L cycles for multiplications and $L + i - 2$ cycles for additions, thus each channel has $L - 1$ free slots for multiplications and $L - i + 1$ free slots for additions (idle states in Fig. 9).

Let us assume, for simplicity, that $k = 3$. The free slots in each MAC(i) will accommodate operations from one MAC(j), $j = \beta + 1, \ldots, 2\beta$, and one MAC($l$), $l = 2\beta + 1, \ldots, 3\beta$. Since each MAC($j$) requires L multiplications and $L + j - 2$ additions and each MAC(l) requires L cycles for multiplications and $L + l - 2$ cycles for additions, the cycles required to accommodate the results are $2L + (j + l) - 4$. The free slots in each MAC(i) are $L - i + 1$ thus the extra cycles to produce all results in each MAC(i) are $2L + (j + l) - 4 - (L - i + 1) = L + (i + j + l) - 5$. The best combinations for (i, j, l) to minimize the quantity $L + (i + j + l) - 5$ can be found using the pseudo-code [44, 48]

```
1: for all $i = 1, \ldots, \beta$ do
2:     for $j = \beta + 1, \ldots, 2\beta$ do
3:         $\alpha_{ijl} \leftarrow L + (i + j + l) - 5 \; \{\forall l \in [2\beta + 1, 3\beta]\}$
4:     end for
5: end for
6: find $\nu = \max(\alpha_{ijl})$ common $\forall i$
7: for all $i = 1, \ldots, \beta$ do
8:     match $i$ with $j$ and $l$, such that $\alpha_{ijl} \leq \nu$
9: end for
```

which calculates all possible values of extra cycles for all combinations of (i, j, l) and in step 6 we select ν as the maximum common value for all MAC(i). For every distinct combination (i, j, l) that satisfies $\alpha_{ijl} < \nu$, we match the corresponding units until all distinct pairs of units in positions (j, l) are assigned to a distinct unit in position i. The remaining 6 steps of the DRAMMM require $6k$ cycles, in total [44, 48].

6 Application to Elliptic Curve Cryptography

In the following, an application of RNSMMM algorithm to ECC is demonstrated. The design employs bases of moduli offering efficient arithmetic and differs from the approaches analyzed before in the sense that dedicated components are employed for IO conversions and BE operations. The design employs Eq. (23) to obtain the MRS digits of the result, while (22) is computed modulo each modulus of the new base.

Sets of three and four moduli are used in implementing the RNSMMM of Algorithm 4. The form of the moduli determines to a large extent the efficiency of the arithmetic operations and the layout of the input/output converters [36]. The RNS bases employed are shown in Table 2. In the first base, RNS moduli of the form $2^r - 2^{t_i} - 1$, where $t_i < r/2$, are employed, which offer simple modulo reduction operations [3]. The second base is realized by sets of three or four moduli of the special forms $\{2^r, 2^{r+1} - 1, 2^r - 1\}$ [34] and $\{2^r, 2^r - 1, 2^{r+1} - 1, 2^{r-1} - 1\}$, which also provide efficient arithmetic operations and IO conversions [3]. In order to use the result of RNSMMM in subsequent modular multiplications, it is required that $4N < M < M'$ [3], which is true for the provided bases.

6.1 *Modular Adders and Multipliers*

For the first base, where moduli of the form $2^r - 2^{t_i} - 1$ are utilized, the modular adder and multiplier depicted in Fig. 13 are employed. Regarding modular multiplication, two r-bit operands are multiplied and a $2r$-bit value is obtained. Modular reduction

Table 2 RNS bases for use in ECC

	Field (bit)	First base $\mathcal{M}$	Second base $\mathcal{M}\simeq$
3 − modulus RNSbases	160	$\{2^{56}-2^{11}-1,$ $2^{56}-2^{16}-1,$ $2^{56}-2^{20}-1\}$	$\{2^{56},$ $2^{56}-1,$ $2^{57}-1\}$
3 − modulus RNSbases	192	$\{2^{66}-2^{17}-1,$ $2^{66}-2^{18}-1,$ $2^{66}-2^{24}-1\}$	$\{2^{66},$ $2^{66}-1,$ $2^{67}-1\}$
4 − modulus RNSbases	192	$\{2^{50}-2^{20}-1,$ $2^{50}-2^{22}-1,$ $2^{50}-2^{18}-1,$ $2^{50}-2^{10}-1\}$	$\{2^{50},$ $2^{50}-1,$ $2^{51}-1,$ $2^{49}-1\}$
4 − modulus RNSbases	224	$\{2^{58}-2^{22}-1,$ $2^{58}-2^{13}-1,$ $2^{58}-2^{10}-1,$ $2^{58}-2^{16}-1\}$	$\{2^{58},$ $2^{58}-1,$ $2^{59}-1,$ $2^{57}-1\}$
4 − modulus RNSbases	256	$\{2^{66}-2^{22}-1,$ $2^{66}-2^{24}-1,$ $2^{66}-2^{18}-1,$ $2^{66}-2^{17}-1\}$	$\{2^{66},$ $2^{66}-1,$ $2^{67}-1,$ $2^{65}-1\}$

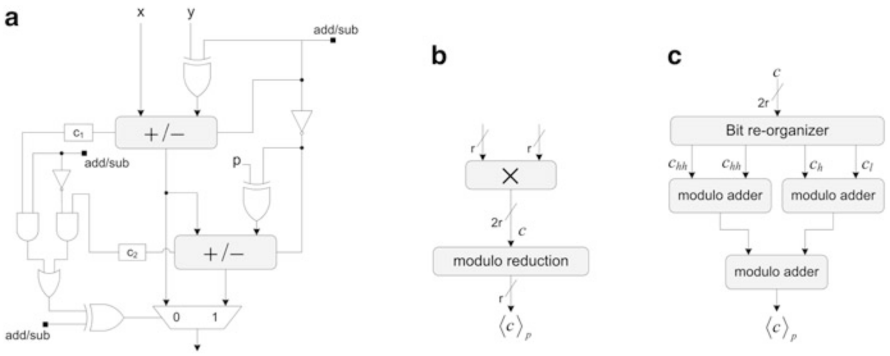


Fig. 13 (**a**) Modulo p adder/subtractor [45]. (**b**) Modulo $2^r-2^{t_i}-1$ multiplier. (**c**) Reduction circuit [13]

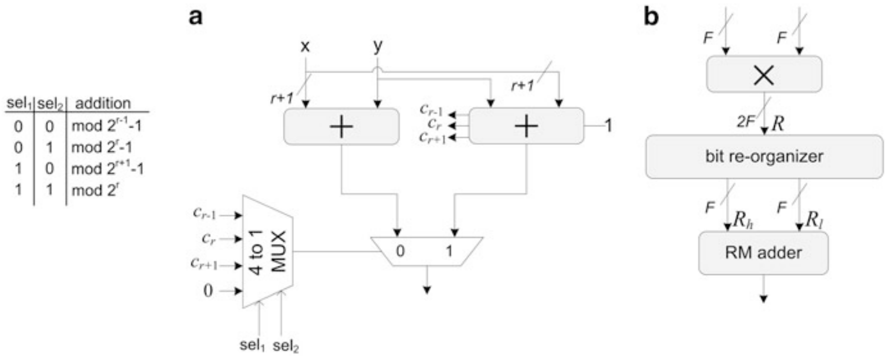
of a $2r$-bit value c with moduli of the form $2^r-2^{t_i}-1$ can be written using its higher r bits, denoted as c_h, and its r lower bits, denoted as c_l, as

$$\langle c\rangle_{2^r-2^{t_i}-1} = \langle c_h 2^r + c_l\rangle_{2^r-2^{t_i}-1}\,. \tag{43}$$

Since $2^r \mod (2^r - 2^{t_i} - 1) = 2^{t_i} + 1$, it holds that

$$
\begin{aligned}
\langle c\rangle_{2^r-2^{t_i}-1} &= \left\langle \langle c_h(2^{t_i}+1)\rangle_{2^r-2^{t_i}-1} + c_l \right\rangle_{2^r-2^{t_i}-1} \\
&= \left\langle \overbrace{\langle c_h 2^{t_i}\rangle_{2^r-2^{t_i}-1}}^{(r+t)-bits} + c_h + c_l \right\rangle_{2^r-2^{t_i}-1} \\
&= \left\langle \langle c_{hh}2^r + c_{hl}\rangle_{2^r-2^{t_i}-1} + c_h + c_l \right\rangle_{2^r-2^{t_i}-1} \\
&= \left\langle \langle c_{hh}2^{t_i} + c_{hh}\rangle_{2^r-2^{t_i}-1} + c_{hl} + c_h + c_l \right\rangle_{2^r-2^{t_i}-1} \\
&= \left\langle c_{hh}\underbrace{0\ldots0}_{t_i-bits} + c_{hh} + c_{hl} + c_h + c_l \right\rangle_{2^r-2^{t_i}-1} \\
&= \left\langle \underbrace{c_{hh}c_{hh}}_{concatenation} + c_{hl} + c_h + c_l \right\rangle_{2^r-2^{t_i}-1}
\end{aligned}
\tag{44}
$$

For the second base, the reconfigurable modular (RM) adder shown in Fig. 14 is employed. Based on this adder, addition and multiplication modulo 2^r, $2^{r-1}-1$, $2^r - 1$, and $2^{r+1} - 1$ can be accommodated in the same hardware module. Note that the RM adder shown in Fig. 14 has $(r-1)$-bit FA delay less than the modulo adder in Fig. 13 (worst case), thus the second base supports more efficient arithmetic operations. Regarding multiplication, the $2F$-bit result R is split into two F-bit LSD and MSD parts (R_l and R_h respectively), where ($F = r, r-1, r+1$) and reduction modulo $2^F - 1$ can be achieved by a modular addition of R_l and R_h [3].



sel_1	sel_2	addition
0	0	mod 2^{r-1}-1
0	1	mod 2^r-1
1	0	mod 2^{r+1}-1
1	1	mod 2^r

Fig. 14 (**a**) Reconfigurable modular (RM) adder, (**b**) RM Multiplier, ($F = r, r-1, r+1$) [13]

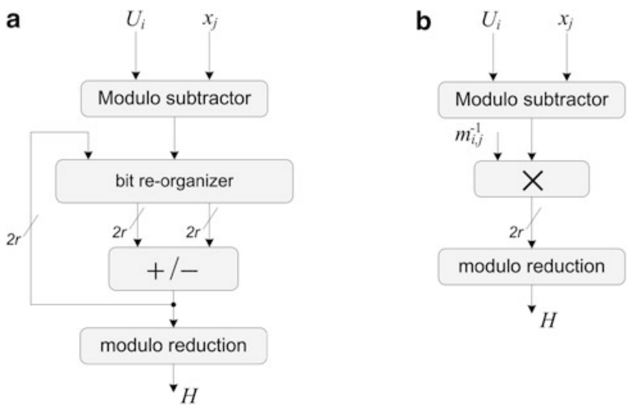


Fig. 15 Calculation of core operation H in RNS-to-MRS conversion for the first base (**a**) area efficient design, (**b**) fast design [13]

6.2 Conversion from Base $\mathcal{M}\simeq$ to Base $\mathcal{M}$

In step 3 of RNSMMM, a base conversion from base $\mathcal{M}\simeq$ to base $\mathcal{M}$ is required, which consists of a residue-to-MRS conversion in base $\mathcal{M}\simeq$ and then an MRS-to-residue conversion in base $\mathcal{M}$. The core operation in calculation of U_i, $\forall i = 2, 3, 4$, in (23) is

$$H = \left\langle \left(x_j - U_i\right) m_{i,j}^{-1} \right\rangle_{m_j} . \tag{45}$$

Hardware implementations of (45) for area and time efficient designs are shown in Fig. 15. Considering four-modulus RNS bases, for each U_i ($i = 2, 3, 4$), the implementation shown in Fig. 15 is employed. The bit re-organizer provides the required shifts according to the pre-calculated multiplicative inverses.

Residues in $\mathcal{M}$ must be calculated after the calculation of mixed radix digits in base $\mathcal{M}\simeq$. In the calculation of MRS-to-RNS conversion from base $\mathcal{M}\simeq$ to base $\mathcal{M}$ for the four-modulus RNS bases, it holds that

$$x_j = \langle U_1 + m_1 \left(U_2 + m_2(U_3 + m_3 U_4)\right)\rangle_{m_j} , \tag{46}$$

where m_j are the moduli of the forms $2^r, 2^r-1, 2^{r+1}-1, 2^{r-1}-1$, and $m_i, i = 1, 2, 3$, are moduli of the form $2^r - 2^{t_i} - 1$. The form of the considered bases with simple multiplicative inverses allows for fast or area-efficient adder-based structures, which can be realized by using one RM adder for each modulus [13].

6.3 Conversion from Base $\mathcal{M}$ to Base $\mathcal{M}_{\simeq}$

In order to mechanize RNS-to-MRS conversion in base $\mathcal{M} = (2^r, 2^r - 1, 2^{r+1} - 1)$, based on (23) and considering $m_1 = 2^r, m_2 = 2^r - 1, m_3 = 2^{r+1} - 1$, we get

$$U_1 = x_1 \tag{47}$$

$$U_2 = \left\langle (x_2 - U_1)\, m_{1,2}^{-1} \right\rangle_{m_2} \tag{48}$$

$$U_3 = \left\langle \left((x_3 - U_1)\, m_{1,3}^{-1} - U_2 \right) m_{2,3}^{-1} \right\rangle_{m_3} . \tag{49}$$

The required multiplicative inverses in (48) and (49) are $\left\langle m_1^{-1} \right\rangle_{m_2} = 1, \left\langle m_1^{-1} \right\rangle_{m_3} = 2$ and $\left\langle m_2^{-1} \right\rangle_{m_3} = -2$ [34]. Due to the simple form of multiplicative inverses, the aforementioned adder-based structure can be employed both for the fast and the area efficient design. Regarding the MRS-to-RNS conversion to base $\mathcal{B}$, it holds that

$$x_j = \langle U_1 + m_1 (U_2 + m_2 (U_3 + m_3 U_4)) \rangle_{m_j} . \tag{50}$$

It is apparent that all calculations in (50) consist of simple shifts and addition operations.

6.4 Hardware Architecture for RNS Montgomery Multiplication

Pipelined RNS architectures for the RNSMMM are shown in Fig. 16. The area efficient design employs one modulo $2^r - 2^{t_i} - 1$ multiplier, one RM multiplier, one RM adder, and two base extension units with adder-based structure, connected in a four-stage pipelined layout (Fig. 16b). The alternative design optimized for high-speed implementations utilizes a six-stage pipelined layout, shown in Fig. 16a. In each modulus channel of stages 1 and 4 of the pipelined implementations, the modular multipliers and adders in Figs. 13 and 14 are employed. For the base extension operations, the modulo adders and multipliers described in respective subsections are utilized.

A complete ECC processor is illustrated in Fig. 17. It consists of IO converters for the conversions to/from RNS representations, a register file, the ALU unit with the presented RNSMMM architectures and the converter from projective to affine coordinates. Based on the control unit a corresponding algorithm for point

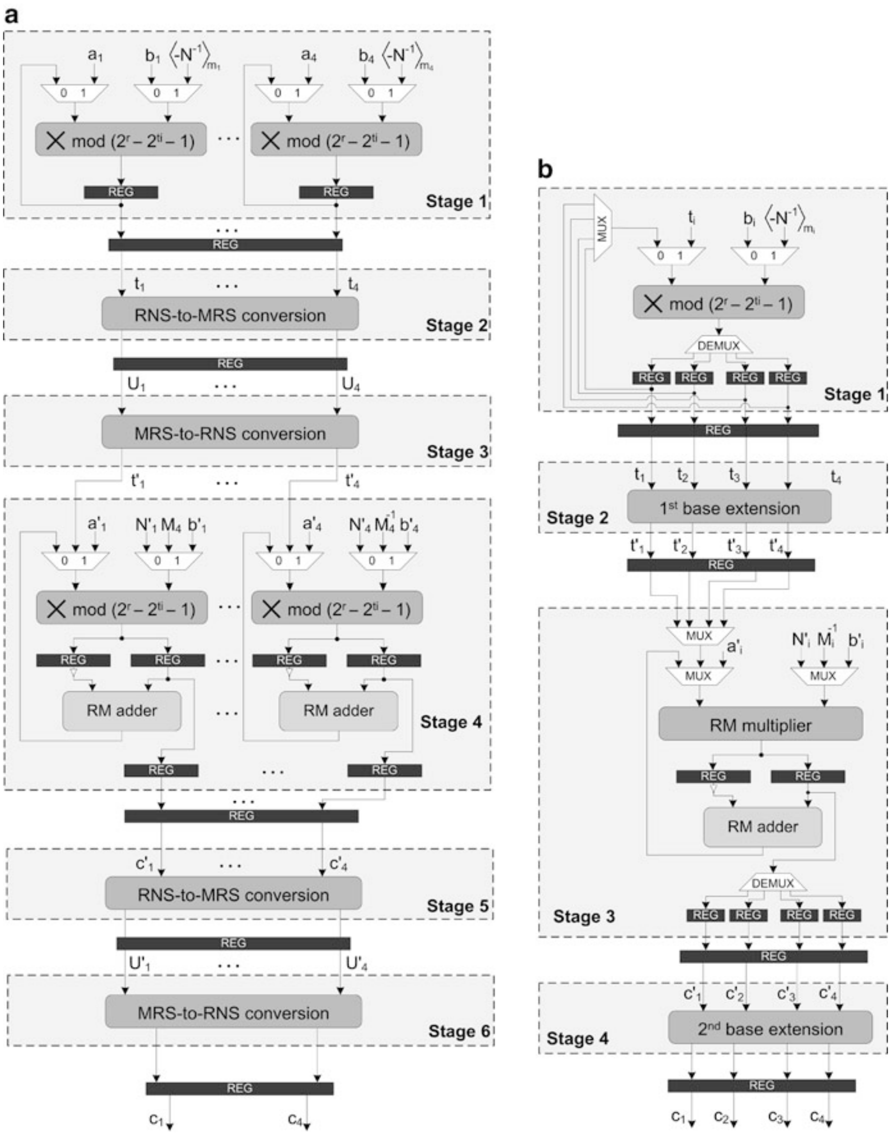
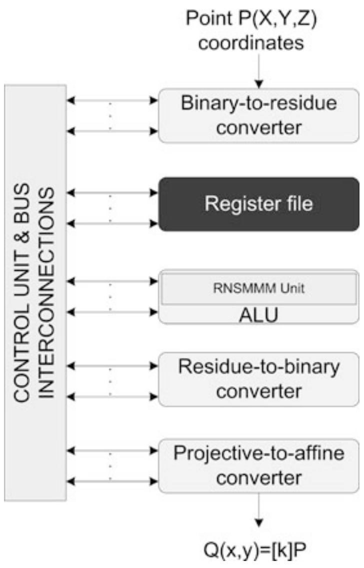


Fig. 16 Pipelined RNSMMM architectures for ECC, (**a**) Fast design, (**b**) Area efficient design

multiplication is performed, like for example the Algorithm 1, based on the binary expansion of scalar [k]. Since the presented bases support efficient arithmetic, the IO converters encompass an architecture similar to the ones presented for the modular addition, subtraction, and base extension.

Fig. 17 General architecture of an ECC processor


Point P(X,Y,Z) coordinates
CONTROL UNIT & BUS INTERCONNECTIONS
Binary-to-residue converter
Register file
RNSMMM Unit
ALU
Residue-to-binary converter
Projective-to-affine converter
Q(x,y)=[k]P

7 Robustness Issues

In an RSA-CRT scheme, the digital signature operation $S = M^d \mod N$ is split in two operations $S_p = M^{d_p} \mod p$ and $S_q = M^{d_q} \mod q$, where $d_p = d \mod$ (p-1) and $d_q = d \mod (q-1)$. CRT ensures that the combination of these two values produces the signature S as

$$S = S_q + \left[\left(S_p - S_q\right) \cdot \left(q^{-1} \mod p\right) \mod p\right] \cdot q \qquad (51)$$

denoted from now on as $S = CRT(S_p, S_q)$ [25]. In this way, an approximate 4-time speedup of operations is achieved [28, 29].

Despite this significant performance improvement, RSA-CRT was proved to be extremely vulnerable against hardware-fault attacks [1, 9, 16]. Assume an erroneous output generated randomly during the execution of a cryptographic operation. Without loss of generality, let the fault be in the modulus p channel, denoted as $\tilde{S}_p$. This will produce a faulty signature $\tilde{S} = CRT\left(\tilde{S}_p, S_q\right)$. An adversary can then factorize the public modulus n by computing its prime factor q as $q = \gcd\left\{\left(\tilde{S}^e - m\right) \mod n, n\right\}$ and consequently obtain $p = n/q$.

In [49], Shamir modified the basic RSA-CRT algorithm in (51) by introducing a random prime r so that $S_{pr} = m^{d \mod (p-1)(r-1)} \mod pr$ and $S_{qr} = m^{d \mod (q-1)(r-1)} \mod qr$. The method checks whether $S_{pr} \equiv S_{qr} \mod r$ holds before combining them with CRT. If $S_{pr} \equiv S_{qr} \mod r$, the computation is error-free, but the step of CRT combination is left unprotected. Moreover, Shamir's method requires the knowledge of the straightforward RSA private key d in an RSA-CRT context, which is unpractical since the key material is given in CRT format [59].

Shamir's countermeasures were broken by exploiting this weakness [1]. The authors proposed an improved implementation that included the protection of the CRT re-combination step. But random number generation is a problem in this scheme, since generating random numbers for each signature operation results in large time overhead.

In [59], the authors proposed a method based on modulus expansion. It computes $m^d \mod n$ in $\mathbb{Z}_{Nr^2}$, where r is a small random integer co-prime with n. The message m is transformed to $\widehat{m}$ such that $\widehat{m} = m \mod n$ and $\widehat{m} = 1 + r \mod r^2$. Then, S and $\widehat{S}$ are computed as $S = m^d \mod n$ and $\widehat{S} = \widehat{m}^d \mod nr^2$. If $\widehat{S} \equiv S \mod n$, then the protocol is error-free. However, the method did not improve much the performance overhead [31].

The Maximum Likelihood Estimation (MLE) algorithm was also exploited as a countermeasure scheme [1]. Unlike the square-and-multiply algorithm which performs on average 1.5 modular multiplications per bit of the exponent, the MLE algorithm performs two modular multiplications for each bit of exponent, and thereby increases execution time.

An ingenious fault-attack based on the *safe-error* concept has been developed in [64]. It was observed that during a modular exponentiation using typical Square and Multiply algorithms, if the exponent bit is 0, then the result of a modular multiplication is not used. By inducing an error during multiplication and by testing whether the result is correct or not, the attacker can deduce the bit of the secret exponent. However, a countermeasure was provided using MLE [21].

Another class of countermeasures is based on "fault-infective" techniques [8, 65]. They are based on the idea of modifying the RSA-CRT computations in such a way that a faulty signature in one channel will infect the final signature after the CRT recombination. Unfortunately, like in [49], not only the knowledge of d is required, but also the techniques rely on some very strong assumptions. For example, some parameters t_1 and t_2 introduced in [8] require that $\gcd(t_1, t_2) = \gcd(d, \varphi(t_1)) = \gcd(d, \varphi(t_2)) = 1$, where φ is the Euler's totient function. t_1, t_2 should normally be generated once, along with the RSA key, and the same values should be used throughout the lifetime of the key. However, these values cannot be stored in such a personalized context, meaning that the generation of t_1, t_2 for each signature is not a task of negligible computational complexity.

The majority of the aforementioned countermeasures are based on modifications of the RSA-CRT protocol, which amount to extra operations and increased algorithmic complexity for the RSA-CRT execution. These solutions rely on the 2-modulus splitting of RSA calculations using a naive RNS consisting of just the moduli p and q. In a different approach, the multi-modulus RNS Montgomery multipliers presented in the previous sections are examined from a hardware-fault tolerance point of view [47], which is examined next.

7.1 *Hardware-Fault Tolerance in MRC-Based RNS Montgomery Multipliers*

It is apparent that steps 1,2,4,5 and 6 of the RNSMMM Algorithm 3 are performed in parallel in each modulus channel. If the algorithm was completely parallel, an error in modulus channel i would not influence the remaining channels and thus the GCD attack would hold. Fortunately, it has been shown that the base extension provides the desired mechanism for fault tolerance [47].

Assume a permanent error $\tilde{t}_i$ in modulus channel $1 \leq i \leq L$. Note that, since step 2 of Algorithm 3 uses the result of step 1, the faulty result will always amount to $\tilde{t}_i$. By observation, employing $\tilde{t}_i$ in the base extension of step 3 yields

$$\tilde{t}_i \leftarrow \left\langle \left(\tilde{t}_i - t_j\right) q_{j,i}^{-1} \right\rangle_{m'_i}, i \in [2, L], \forall j \in [1, i-1]. \tag{52}$$

Equation (52) corresponds to the steps 1–7 of Algorithm 15 and implies that an error occurred in position i will always cascade to the other channels and produce a faulty $\tilde{t}_L$, even if the error occurs at the very last step of calculations in channel L. This value is used in step 9 to continue the base extension process. An important observation is that at this step the faulty $\tilde{t}_L$ is injected to all channels according to

$$\tilde{t}'_i \leftarrow \langle \tilde{t}_L \rangle_{m_i}, \forall i \in [1, L] \tag{53}$$

and similarly the faulty $\tilde{t}'_i$s produce

$$\tilde{t}'_i \leftarrow \left\langle \tilde{t}'_i \cdot m'_j + \tilde{t}_j \right\rangle_{m_i}, \forall i \in [1, L], \forall j \in [1, L-1]. \tag{54}$$

As a result, a faulty $\tilde{t}_{\mathcal{M}}$ is generated at step 3 of Algorithm 15 and injected in step 4 for subsequent calculations. Note that due to (53), it is assured that all channels after the first base extension will be infected. Using a similar analysis, it is easy to show that even if the error occurs after the first base extension, the second base extension at step 7 of Algorithm 3 will infect all channels in the same manner, thus making the GCD attack infeasible [47].

A special case is when the error is not permanent and is inserted in a channel $i, i \in [1, L]$, during the base extension. If the error is generated during steps 1–7 of Algorithm 15, step 9 will inject the error to all other channels, according to (53). The case that an error is inserted in channel $i, i \in [1, L]$, during step 11 of Algorithm 15 should also be examined. Although step 11 is executed in parallel for all channels, each channel calculation reuses the results from all other channels. This is also apparent from the recursive form of (53). Due to this, all channels are affected, making GCD attack infeasible. A similar analysis may be conducted for the MRC-based BE in Algorithm 11 [47].

Algorithm 15 MRC-based base extension

```
Input: $x_{\mathcal{M}\simeq} = (x_1, x_2, \ldots, x_L)$
Output: $x_{\mathcal{M}} = (x'_1, x'_2, \ldots, x'_L)$
 1: $U_1 \leftarrow x_1$
 2: for all $i = 2, \ldots, L$ do
 3:     $U_i \leftarrow x_i$
 4:     for $j = 1$ to $i - 1$ do
 5:         $U_i \leftarrow \left\langle (U_i - U_j)\, q_{j,i}^{-1} \right\rangle_{m'_i}$
 6:     end for
 7: end for
 8: for all $i = 1, \ldots, L$ do
 9:     $x'_i \leftarrow \langle U_L \rangle_{m_i}$
10:     for $j = L - 1$ to $1$ do
11:         $x'_i \leftarrow \left\langle x'_i m'_j + U_j \right\rangle_{m_i}$
12:     end for
13: end for
```

7.2 Hardware-Fault Tolerance in CRT-Based RNS Montgomery Multipliers

In [24], the first practical and efficient implementation of RNS Montgomery multiplier based on CRT was presented. The CRT-based algorithm for RNSMMM is identical to Algorithm 3, thus only the BE is represented below as Algorithm 2. Clearly, steps 1–5 and 14–16 of Algorithm 2 involve completely parallel operations in all channels, so fault-tolerance should be examined for the steps 6–13. In the case of a permanent error, a faulty $\tilde{\gamma}_j^*, j \in [1, L]$, is generated in steps 8–9, which consequently produces

$$\tilde{\delta}_{i,j} = \delta_{i,(j-1)} + \xi_j \cdot \left\langle M'_j \right\rangle_{m_i} + \tilde{\gamma}_j^* \cdot \left\langle (-M') \right\rangle_{m_i}, \forall i, j \in [1, L]. \tag{55}$$

This means that all channels are affected by the error, thus the parallel operations of steps 14–16 are also affected.

However, if an adversary is able to insert an error during the steps 14-16, only one (or several) channels can be affected, which makes the GCD attack easily mountable. To overcome this issue, an extra checking procedure could be inserted in steps 14–16 of Algorithm 2 based on the following pseudo code

```
1: for all $i = 1, \ldots, L$ do
2:     if $\delta_{i,L} == \delta_{i,L}$ of step 11 then
3:         $x'_i = \langle \delta_{i,L} \rangle_{m_i}$
4:     else
5:         error detected
6:     end if
7: end for
```

The solution checks whether the quantities $\delta_{i,L}$ are identical to the values obtained in the previous step 11, and, if not, a malicious error has been detected. The solution requires the storage of the L values of step 11 and a comparison with the $\delta_{i,L}$s employed in step 15. Note that this solution does not issue significant overhead, since the checking procedure can be executed only once at the end of an RSA exponentiation.

7.3 Performance

Presenting performance metrics of RNS Montgomery multipliers is out of scope of this section. We have already presented comparative studies in the previous sections as well as in [48], while trade-offs between state-of-the art RNS solutions appear in [15]. There is, however, an important derivative of the presented hardware-fault analysis on RNS Montgomery multipliers. As described in the previous paragraphs, current countermeasures appearing in the literature provide immunity at the cost of extra operations or checking procedures in the RSA-CRT protocol itself, thus the 4-time speedup offered by the use of RSA-CRT is somehow sacrificed to achieve tolerance against hardware-fault attacks.

The presented analysis shows that if RNS Montgomery multipliers are employed instead of non-RNS ones in crypto-hardware design, security is offered for free, with no need for extra checking procedures or modifications to the RSA-CRT protocol as in [1, 8, 16, 31, 49, 59, 64, 65]. At the same time, since immunity comes for free, the 4-time speedup between RSA and RSA-CRT is maintained.

8 Summary and Research Outlook

Being exposed in an unprecedented number of threats and frauds, safe connectivity for all network-based systems has become a predicate necessity. Cryptographic hardware plays a dominant role in the implementation of systems that could offer the desired levels of security. The prospective crypto-hardware designer should not only care for performance but also resistance against attacks. Under this perspective, cryptographic hardware design poses extra difficulties and challenges considering especially the fact that, as years pass by, the security standards need to be constantly strengthened.

This chapter attempted to approach the problem of cryptographic hardware design with the use of unconventional arithmetic in a holistic manner, covering aspects of algorithmic analysis, crypto-hardware design, and security validation of the presented architectures.

An important class of algorithms, namely the RNSMMM and PRNSMMM algorithms were presented for the case of RNS incorporation in $GF(p)$ and PRNS incorporation in $GF(2^n)$ calculations respectively. The most important features and characteristics of these algorithms were analyzed and new, improved versions for both algorithms were presented that support operations in both fields. An extensively analyzed application example for ECC was also presented.

An important security property of the residue arithmetic architectures was also revealed. It was shown that the use of a well-designed, residue-arithmetic, Montgomery multiplier overcomes hardware-fault attack threats, with no need to alter the basic RSA-CRT protocol, while, at the same time, the speed-gains offered by RSA-CRT are maintained.

New parallelization prospects offered by state-of-the-art multi-processor systems could also be investigated. A possible scenario could be that parallel processors perform parallel multiplications on different data-sets of a single message. In such cases, the existence of equivalencies between a serial and a parallel algorithm, if any, should be mathematically proven. Also, in case parallelization is possible, any required algorithmic overhead should be carefully determined and assessed for performance impact. There are various design issues when it comes to multi-processor system design. A careful examination of the impact of interconnections, system I/O delays, etc., on the system's performance and area should be carried out. It should be also considered that these systems require full availability of all input data beforehand, which does not allow for real-time encryption/signing.

Finally, the cryptanalytic properties of RNS-based architectures can also be further extended, to include attacks other than hardware-fault related. The role of BE operation should be meticulously analyzed to reveal new possibilities for cryptanalytic resistance. An interesting derivative is the security potential offered by the presented versatile architectures, by means of changing seamlessly the underlying cryptographic protocols during an established communication channel. Investigating the applicability of RNS to other PKC systems, like for example the emerging lattice-based cryptography [17, 41], could also generate new and interesting cryptanalytic properties, architectures, and algorithms.

In general, current solutions employing conventional binary arithmetic for modular multiplication are based on Montgomery's algorithm (systolic, semi-systolic, etc). These architectures have been extensively analyzed and the optimizations proposed are so fine-grained, that the research space on the field steadily narrows. On the other hand, this chapter provided solid indications that non-conventional arithmetic like RNS and PRNS may provide new means for tackling design problems of crypto-hardware and further extend the research space in this active field.

References

1. Aumüller, C., Bier, P., Fischer, W., Hofreiter, P., Seifert, J.P.: Fault attacks on RSA with CRT: concrete results and practical counter-measures. In: Proceedings of International Workshop Cryptographic Hardware and Embedded Systems (CHES '02), pp. 260–275, 2002
2. Bajard, J., Eynard, J., Gandino, F.: Fault detection in rns montgomery modular multiplication. In: 21st IEEE Symposium on Computer Arithmetic, ARITH, pp. 119–126, 2013

3. Bajard, J., Kaihara, M., Plantard, T.: Selected RNS Bases for Modular Multiplication. In: 19th IEEE International Symposium on Computer Arithmetic, pp. 25–32, 2009
4. Bajard, J.C., Didier, L.S., Kornerup, P.: Modular multiplication and base extensions in residue number systems. In: Proceedings of the 15th Symposium on Computer Arithmetic, ARITH '01, pp. 59–65, 2001
5. Bajard, J.C., Imbert, L.: A full RNS implementation of RSA. IEEE Trans. Comput. **53**, 769–774 (2004)
6. Bajard, J.C., Imbert, L., Jullien, G.A.: Parallel Montgomery multiplication in $GF(2^k)$ using trinomial residue arithmetic. IEEE Symp. Comput. Arith. **0**, 164–171 (2005). doi:10.1109/ARITH.2005.34
7. Blake, I., Seroussi, G., Smart, N.: Elliptic Curves in Cryptography. Cambridge University Press, Cambridge (2002)
8. Blömer, J., Otto, M., Seifert, J.P.: A new CRT-RSA algorithm secure against bellcore attacks. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS '03, pp. 311–320, 2003
9. Boneh, D., DeMillo, R., Lipton, R.: On the importance of eliminating errors in cryptographic computations. J. Cryptol. **14**, 101–119 (2001)
10. Deschamps, J.P., Bioul, G.J.A., Sutter, G.D.: Synthesis of Arithmetic Circuits: FPGA, ASIC and Embedded Systems. Wiley, Hoboken, New Jersey (2006)
11. Di Claudio, E.D., Piazza, F., Orlandi, G.: Fast combinatorial rns processors for dsp applications. IEEE Trans. Comput. **44**(5), 624–633 (1995)
12. Ercegovac, M., Lang, T.: Digital Arithmetic. Morgan Kaufmann, San Francisco (2004)
13. Esmaeildoust, M., Schinianakis, D., Javashi, H., Stouraitis, T., Navi, K.: Efficient RNS Implementation of Elliptic Curve Point Multiplication Over $GF(p)$. IEEE Trans. Very Large Scale Integr. VLSI Syst. **8**(21), 1545–1549 (2013)
14. Gandino, F., Lamberti, F., Montuschi, P., Bajard, J.: A General Approach for Improving RNS Montgomery Exponentiation Using Pre-processing. In: 20th IEEE Symposium on Computer Arithmetic, ARITH, pp. 195–204, 2011
15. Gandino, F., Lamberti, F., Paravati, G., Bajard, J.C., Montuschi, P.: An algorithmic and architectural study on Montgomery exponentiation in RNS. IEEE Trans. Comput. **61**(8), 1071–1083 (2012)
16. Giraud, C.: An RSA implementation resistant to fault attacks and to simple power analysis. IEEE Trans. Comput. **55**(9), 1116–1120 (2006)
17. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski, B.J. (ed.) Advances in Cryptology CRYPTO '97. Lecture Notes in Computer Science, vol. 1294, pp. 112–131. Springer Berlin/Heidelberg (1997). doi:10.1007/BFb0052231. http://dx.doi.org/10.1007/BFb0052231
18. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curves Cryptography. Springer and Hall/CRC, New York (2004)
19. Hiasat, A., Abdel-Aty-Zohdy, H.: A high-speed division algorithm for residue number system. In: IEEE International Symposium on Circuits and Systems, 1995. ISCAS '95, vol. 3, pp. 1996–1999, 1995
20. Huang, C.H., Taylor, F.J.: A memory compression scheme for modular arithmetic. IEEE Trans. Acoust. Speech Signal Process. **ASSP-27**, 608–611 (1979)
21. Joye, M., Yen, S.M.: The Montgomery powering ladder. In: Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES'02) LNCS, pp. 291–302, 2002
22. Jullien, G.A.: Residue number scaling and other operations using rom arrays. IEEE Trans. Comput. **C-27**(4), 325–336 (1978)
23. Kaliski, B.: TWIRL and RSA key size. http://www.rsasecurity.com/rsalabs/node.asp?id=2004
24. Kawamura, S., Koike, M., Sano, F., Shimbo, A.: Cox-Rower architecture for fast parallel Montgomery multiplication. In: EUROCRYPT'00: Proceedings of the 19th international conference on Theory and application of cryptographic techniques, pp. 523–538. Springer, Berlin/Heidelberg (2000)

25. Knuth, D.E.: The Art of Computer Programming : Seminumerical Algorithms, 3rd edn. vol. 2. Addison-Wesley Longman Publishing, Boston, MA (1997)
26. Koblitz, N.: Elliptic curve cryptosystems. Math. Comput. **48**, 203–209 (1987)
27. Koren, I.: Computer Arithmetic Algorithms. A K Peters, Natick, Massachusetts (2002)
28. Lab, R.: High-speed RSA implementation (2011). ftp://ftp.rsasecurity.com/pub/pdfs/tr201.pdf
29. Lab, R.: RSA hardware implementation (2011). ftp://ftp.rsasecurity.com/pub/pdfs/tr801.pdf
30. Lidl, R., Niederreiter, H.: Introduction to Finite Fields and their Applications. Cambridge University Press, New York (1986)
31. Ma, K., Liang, H., Wu, K.: Homomorphic property-based concurrent error detection of RSA: a countermeasure to fault attack. IEEE Trans. Comput. **61**(7), 1040–1049 (2012)
32. McEliece, R.J.: Finite Field for Scientists and Engineers. Kluwer Academic, Norwell (1987)
33. Miller, V.: Use of elliptic curves in cryptography. In: Advances in Cryptology (CRYPTO'85) LNCS, vol. 218, pp. 47–426, 1986
34. Mohan, P.: RNS-to-binary converter for a new three-moduli Set $\{2^{n+1} - 1, 2^n, 2^n - 1\}$. IEEE Trans. Circuits Syst. Express Briefs **54**(9), 775–779 (2007)
35. Montgomery, P.L.: Modular multiplication without trial division. Math. Comput. **16**, 519–521 (1985)
36. Navi, K., Molahosseini, A., Esmaeildoust, M.: How to teach residue number system to computer scientists and engineers. IEEE Trans. Educ. **54**(1), 156–163 (2011)
37. Nozaki, H., Motoyama, M., Shimbo, A., Kawamura, S.: Implementation of RSA algorithm Based on RNS Montgomery multiplication. In: Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES'01) LNCS, vol. 2162, pp. 364–376, 2001
38. Posch, K., Posch, R.: Base extension using a convolution sum in residue number systems. Computing **50**(2), 93–104 (1993)
39. Posch, K., Posch, R.: Modulo reduction in residue number systems. Trans. Parallel Distrib. Syst. **6**(5), 449–454 (1995)
40. Ramirez, J., Fernandez, P., Meyer-Base, U., Taylor, F., Garcia, A., Lloris, A.: Index-based rns dwt architectures for custom ic designs. In: IEEE Workshop on Signal Processing Systems, pp. 70–79, 2001
41. Regev, O.: Lattice-based cryptography. In: Advances in Cryptology CRYPTO '06. Lecture Notes in Computer Science, pp. 131–141. Springer, Berlin/Heidelberg (2006)
42. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commum. ACM **21**, 120–126 (1978)
43. Savaş, E., Tenca, A., Koç, Çetin.: A scalable and unified multiplier architecture for finite fields $GF(p)$ and $GF(2^m)$. In: Cryptographic Hardware and Embedded Systems (CHES 2000). Lecture Notes in Computer Science, vol. 1965, pp. 277–292. Springer, Berlin (2000)
44. Schinianakis, D.: Versatile architectures for cryptographic systems, Ph.D. dissertation. University of Patras, Patras, Greece, 2013
45. Schinianakis, D., Fournaris, A., Michail, H., Kakarountas, A., Stouraitis, T.: An RNS implementation of an F_p elliptic curve point multiplier. IEEE Trans. Circuits Syst. I **56**(6), 1202–1213 (2009)
46. Schinianakis, D., Stouraitis, T.: A RNS Montgomery multiplication architecture. In: IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1167–1170, 2011
47. Schinianakis, D., Stouraitis, T.: Hardware-fault attack handling in RNS-based Montgomery multipliers. In: IEEE International Symposium on Circuits and Systems (ISCAS), pp. 3042–3045, 2013
48. Schinianakis, D., Stouraitis, T.: Multifunction residue architectures for cryptography. IEEE Trans. Circuits Syst. I: Reg. pap. **61(4)**, 1156–1169 (2014)
49. Shamir, A.: Improved method and apparatus for protecting public key schemes from timing and fault attacks. U.S Patent, 1999
50. Shenoy, M., Kumaresan, R.: A fast and accurate RNS scaling technique for high speed signal processing. IEEE Trans. Acoust. Speech Signal Process. **37**(6), 929–937 (1989)

51. Skavantzos, A., Wang, Y.: New efficient rns-to-weighted decoders for conjugate-pair-moduli residue number systems. In: Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers, vol. 2, pp. 1345–1350, 1999
52. Smith, W.: Swift. In: Symposium on Very High Speed Computing Technology (held with IEEE ICASSD Conference) 1980
53. Sousa, L.: Efficient method for magnitude comparison in RNS based on two pairs of conjugate moduli. In: 18th IEEE Symposium on Computer Arithmetic, 2007. ARITH '07, pp. 240–250, 2007
54. Szabo, N., Tanaka, R.: Residue Arithmetic and its Applications to Computer Technology. McGraw-Hill, New York (1967)
55. Taylor, F., Zelniker, G., Smith, J., Mellott, J.: The gauss machine-a dsp processor with a high rns content. In: International Conference on Acoustics, Speech, and Signal Processing, 1991. ICASSP-91, vol. 2, pp. 1081–1084, 1991
56. Taylor, F.J.: A vlsi residue arithmetic multiplier. IEEE Trans. Comput. **C-31**(6), 540–546 (1982)
57. Taylor, F.J.: Residue arithmetic: a tutorial with examples. IEEE Comput. **17**, 50–62 (1988)
58. Tomczak, T.: Fast sign detection for RNS $(2^n-1, 2^n, 2^n+1)$. IEEE Trans. Circuits Syst. Regul. Pap. **55**(6), 1502–1511 (2008)
59. Vigilant, D.: RSA with CRT: a new cost-effective solution to Thwart fault attacks. In: Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems (CHES 08), pp. 130–145, 2008
60. Wang, W., Swamy, M., Ahmad, O., Wang, Y.: New Chinese Remainder Theorems applications to special moduli sets. In: CCECE99, vol. 2, pp. 1345–1350, 1999
61. Wang, Y.: Residue-to-binary converters based on new chinese remainder theorems. IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process. **47**(3), 197–205 (2000)
62. Yang, J.H., Chang, C.C., Chen, C.Y.: A high-speed division algorithm in residue number system using parity-checking technique. Int. J. Comput. Math. **81**(6), 775–780 (2004)
63. Yassine, H.M., Moore, W.: Improved mixed-radix conversion for residue number system architectures. IEE Proc. G Circuits Devices Syst. **138**(1), 120–124 (1991)
64. Yen, S., Joye, M.: Checking before output may not be enough against fault-based cryptanalysis. IEEE Trans. Comput. **49**(9), 967–970 (2000)
65. Yen, S., Kim, S., Lim, S., Moon, S.: RSA speedup with Chinese remainder theorem immune against hardware fault cryptanalysis. IEEE Trans. Comput. **52**(4), 461–472 (2003)

Fault Attacks on AES and Their Countermeasures

Subidh Ali, Xiaofei Guo, Ramesh Karri, and Debdeep Mukhopadhyay

Abstract Fault Attacks exploit malicious or accidental faults injected during the computation of a cryptographic algorithm. Combining the seminal idea by Boneh, DeMillo and Lipton with Differential Cryptanalysis, a new field of Differential Fault Attacks (DFA) has emerged. DFA has shown that several ciphers can be compromised if the faults can be suitably controlled. DFA is not restricted to old ciphers, but can be a powerful attack vector even for modern ciphers, like the Advanced Encryption Standard (AES). In this book chapter, we present an overview on the history of fault attacks and their general principle. The chapter subsequently concentrates on the AES algorithm and explains the developed fault attacks. The chapter covers the entire range of attacks finally showing that a single random byte fault can reduce the AES key to 2^8 values, with a time complexity of 2^{30}. Further extensions of the fault attack to multiple byte fault models and attacks targeting the AES key schedule are also presented in the chapter. These attacks emphasize the requirement of counter-measures to detect the underlying faults and accordingly suppress the invalid output. The chapter then presents a survey of existing DFA countermeasures, concluding with the efficient Concurrent Error Detection (CED) schemes which have been developed utilizing the invariance properties in AES. Such a strategy provides near 100 % fault coverage at a less overhead. The combined chapter shows that DFA against AES are practical, and can be prevented using suitable techniques.

The authors "Subidh Ali" and "Xiaofei Guo" have equally contributed to this book chapter.

S. Ali
New York University Abu Dhabi, Abu Dhabi 129188, UAE
e-mail: subidh.ali@nyu.edu

X. Guo • R. Karri
Five MetroTech Center Brooklyn, New York University School of Engineering,
Brooklyn, NY 11201, USA
e-mail: xg243@nyu.edu;rkarri@nyu.edu

D. Mukhopadhyay (✉)
Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur,
Kharagpur, West Bengal 721302, India
e-mail: debdeep@cse.iitkgp.ernet.in

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_5

1 Introduction: Faults and Cryptosystems

The growing complexity of the cryptographic algorithms and the increasing applications of ciphers in real time applications has lead to research in the development of high speed hardware designs or optimized cryptographic libraries for these algorithms. The complex operations performed in these designs and the large state space involved indicates that a complete verification is ruled out. Hence these designs have a chance of being fault prone. Apart from these unintentional faults, faults can also be injected intentionally. Literature shows several ways of fault injection: accidental variation in operating conditions, like voltage, clock frequency, or focussed laser beams in hardware. Software programs can also be subjected to situations, like missing of certain instructions to inflict faults. Apart from the general issue of fault tolerance in any large design, faults bring a complete new aspect when dealing with cryptographic algorithms: security.

The first thing that comes to mind is the relation between faults and secrets. In this section, we first attempt to motivate the impact of faults in the leakage of information.

Motivating Example Consider a pedagogical example comprising of two hardware devices as illustrated in Fig. 1.

The first device has a register storing the values $R1 = (6,0)$, and computes the product $y_{left} = (6,0) \times (a,b)^T = 6a \bmod m$. The value of m is fixed as say 8. The second device on the other hand has the register with value $R2 = (0,2)$ and computes $y_{right} = (0,2) \times (a,b)^T = 2b \bmod m$. The users can feed in values of (a,b), st. $(a,b) \in \{2,6\} \times \{2,6\}$. For the rest of the discussion all the computations are mod 8 and are not explicitly stated.

The user can only input the values (a,b) chosen from the 4 values of $\{2,6\} \times \{2,6\}$. On receiving the inputs (a,b) both the hardwares compute the values of y_{left} and y_{right}. However the user is given either y_{left} or y_{right}, chosen on the basis of a random toss of an unbiased coin which is hidden from the user. The challenge of the user is to guess the outcome of the random coin with a probability better than $\frac{1}{2}$.

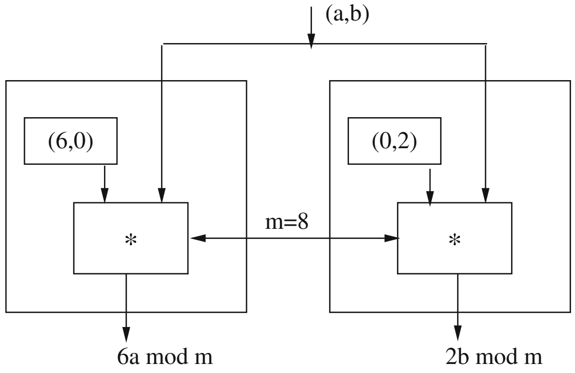


Fig. 1 Effect of faults on secrets

The user is allowed to make multiple queries by providing any of the 4 inputs (a, b). It can be assumed that the random choice is kept constant for all the inputs.

It may be easily observed that $y_{left} = y_{right}$ for all the 4 values of (a, b) which implies that the output y_{left} or y_{right} does not reveal which output is chosen by the random toss. For all the input values of (a, b) the output is 4.

Now consider that one of the hardwares is subjected to a permanent stress, which creates a fault in either the registers $R1$ or $R2$. Hence, either $R1 = r \neq 6$ or $R2 = r \neq 2$. If the fault occurs in $R1$, $y'_{left} = ra$, while $y_{right} = 2b$. Else if the fault occurs in $R2$, $y_{left} = 6a$, while $y'_{right} = rb$. WLOG. assume that the fault is in the first device.

Now the attacker provides two inputs: $(2, 2)$ and $(6, 6)$ to the hardware devices. The attacker observes both of the outputs. If both the outputs are the same then the attacker concludes that the right output is chosen, while if they are different the left output is chosen with probability 1.

Thus this simple example shows that a fault can leak information which seemed to be perfectly hidden in the original design. Thus apart from the malfunction of hardware or software designs, algorithms which hide information (like ciphers) should be analyzed w.r.t. faults. Next, we consider a more non-trivial example of fault based analysis of the popular RSA cryptosystem.

1.1 Fault Analysis of the RSA Cipher

The first fault based attack was mounted on the well-known public key cryptosystem *RSA*. We know that *RSA* works by considering two keys: a *public key* is known to every one, while a *private key* is *secret*. Encryption of a message is performed using the public key, but decryption requires the knowledge of the private key. All the operations are done mod n, where n is the product of two large distinct prime number p and q. The values of p and q are however private and hence not disclosed to all. The encryption key, which is public is a value b, where $1 \leq b \leq \phi(n)$ where $\phi(n)$ is the Euler-Totient function. The decryption key is a private value a, which is selected such that $ab \equiv 1 \, mod \, \phi(n)$. The owner of the private key (p, q, a) publishes the value (b, n) which is the public key.

The encryptor chooses a message x, where $x \in Z_n$. It may be mentioned that $Z_n = \{0, 1, \ldots, n-1\}$. The encryption process is computing the cipher as $y \equiv x^b \, mod \, n$ using the public key b. Since the decryptor knows the value of a, which is the private key, he computes the value of x from y by computing $y^a \equiv (x^b)^a \, mod \, n \equiv x \, mod \, n$. The security of *RSA* is based on the assumption that decryption can be performed only by the knowledge of the private key b. However to obtain the private information from the public value a requires one to compute the modular inverse of a modulo $\phi(n)$. It is believed that to obtain $\phi(n)$ from n requires the knowledge of the prime factors of n, namely p and q. The security of RSA is thus based on the hardness assumption of factorization of large n.

However we explain that under situation of faulty computations the value of the secret exponent a can be retrieved by efficient algorithms. In the attack it is assumed that the attacker is in possession of certain number of plaintext and ciphertext pairs. The attacker has the ability to flip one bit of the value a during computation. Say, the ith bit a_i of a is flipped and modified to $\hat{a}_i$, where $0 \leq i \leq |a|$ and $|a|$ is the bit-length of a. The attacker has access to both fault-free X and faulty plaintexts $\hat{X}$. Therefore, he can compute, $\frac{X}{\hat{X}} = \frac{Y^{\hat{a}}}{Y^{a}} = \frac{Y^{2^i \hat{a}_i}}{Y^{2^i a_i}} \, mod\, n$. If the ratio is equal to Y^{2^i}, the attacker can be sure that $a_i = 0$. On the other hand if the ratio is $\frac{1}{Y^{2^i}}$, the attacker ascertains that $a_i = 1$. The same technique is repeated for all the values of i, thus a can be retrieved. The attack is also applicable when the fault is induced in Y. It is also possible in cases when the fault flips two or more bits. The details are left to the reader as an exercise.

These attacks show that fault analysis can be a powerful tool for attacking ciphers. Significant research has been performed in the field of fault based crypt-analysis of various ciphers of different types. From the seminal paper of [10] fault attacks have been improved with the ideas of differential analysis to attack block ciphers, like Data Encryption Standard (DES). However, after the acceptance of the 128-bit version of the *Rijndael* block cipher, designed by Vincent Rijmen and Joan Daemen, as the Advanced Encryption Standard (AES) in 2001, the main focus of fault attacks have been AES. In the following section we present an overview on the AES algorithm.

2 Preliminaries

The chapter focuses on AES and its fault analysis. The present section provides a top level description of the block cipher algorithm.

2.1 AES Algorithm

AES is an iterative block cipher, designed by Vincent Rijmen and Joan Daemen. The algorithm, originally designed to support both block and key lengths of 128, 192, and 256 bits, the standardizes AES supports only block length of 128 bits, though the key can be of all the three specifications: 128, 192 and 256 bits.

The AES algorithm is an iterated block cipher, meaning the plaintext is applied over several rounds to obtain the final ciphertext. The three versions of AES, ie. AES-128, AES-192, and AES-256 has 10, 12, and 14 rounds. The 128-bit input plaintext is transformed by the rounds into a 128-bit output ciphertext. All the rounds of AES are identical except the last round with a slight change.

Each round of AES encryption consists of SubBytes, ShiftRows, MixColumns, and AddRoundKey denoted by B, S, M, and A, respectively, as shown in Fig. 2.

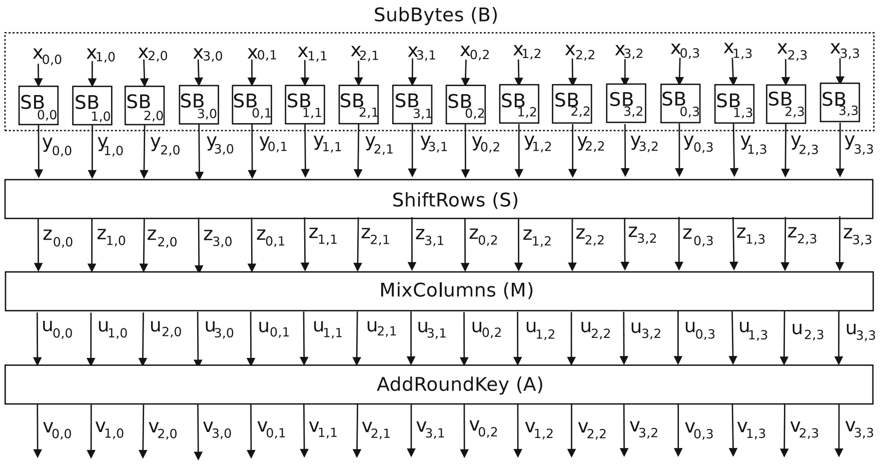


Fig. 2 One AES encryption round

The plaintext is first *mixed* with the input key through a key XORing operation. Subsequently the rounds are applied, which are composed of the above mentioned four operations, B, S, M, and A. The last round is only distinct as MixColumns is not performed.

2.2 Round Transformations of AES

Each operation in every round acts on a 128-bit input ***state***, where each state element is a byte in $GF(2^8)$. Each byte is denoted by $\boldsymbol{s_{r,c}}$ $(0 \leq r, c \leq 3)$ indicating that this byte is in row $\boldsymbol{r}$ and column $\boldsymbol{c}$ in the state matrix.

$$S = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = [s_{r,c}]_{r,c=0}^{3} \tag{1}$$

In SubBytes, all bytes are processed separately by 16 S-boxes (SBs in Fig. 2). Each SB performs a nonlinear transformation of the input byte. If X is the input, the output is:

$$Y = B(X) = [x_{r,c}]_{r,c=0}^{3} \tag{2}$$

In ShiftRows, the rows of the state are shifted cyclically byte-wise using a different offset for each row. Row 0 is not shifted, while rows 1, 2, and 3 are cyclically shifted to the left by 1, 2, and 3 bytes respectively. The resulting output is:

$$Z = S(Y) = \begin{bmatrix} y_{0,0} & y_{0,1} & y_{0,2} & y_{0,3} \\ y_{1,1} & y_{1,2} & y_{1,3} & y_{1,0} \\ y_{2,2} & y_{2,3} & y_{2,0} & y_{2,1} \\ y_{3,3} & y_{3,0} & y_{3,1} & y_{3,2} \end{bmatrix} = [y_{r,(r+c)\ mod\ 4}]_{r,c=0}^{3} = [z_{r,c}]_{r,c=0}^{3} \tag{3}$$

In MixColumns, the output state is obtained by multiplying the output of ShiftRows by a constant matrix. The resulting output is:

$$U = M(Z) = [u_{r,c}]_{r,c=0}^{3} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} z_{0,0} & z_{0,1} & z_{0,2} & z_{0,3} \\ z_{1,0} & z_{1,1} & z_{1,2} & z_{1,3} \\ z_{2,0} & z_{2,1} & z_{2,2} & z_{2,3} \\ z_{3,0} & z_{3,1} & z_{3,2} & z_{3,3} \end{bmatrix} \tag{4}$$

In AddRoundKey, the round key $K = [k_{r,c}]_{r,c=0}^{3}$ is added (modulo-2) to the 128-bit state U. The resulting round output is:

$$V = A(K, U) = [k_{r,c}]_{r,c=0}^{3} + [u_{r,c}]_{r,c=0}^{3} = [v_{r,c}]_{r,c=0}^{3} \tag{5}$$

2.3 Key Scheduling Algorithm

The round keys are generated by the AES key scheduling algorithm, as shown in Algorithm 1. The master key K is used to derive all the round keys, where N_k, N_r and K^r represent the key length in words (4 bytes), number of rounds and the rth round key respectively. The input key is of size $4N_k$ bytes. The algorithm produces the rth round key, which is denoted by K^r. As $0 \leq r \leq N_r$, the total expanded round keys can be stored in the vector $W[N_b(N_r+1)]$, where N_b is the block length of AES. The algorithm consists of operations: *SubWord* and *RotWord*, which are explained as follows: The operation SubWord consists of SubByte operations applied to each of the 4 bytes separately on every byte of a word. The RotWord operation is a cyclic circular left shift on the bytes of an input word. Finally, the round constant abbreviated as $Rcon[n] = (\{02\}^n, \{00\}, \{00\}, \{00\})$. For more details one can refer to the AES specification [43].

3 Introduction to Differential Fault Analysis

The first fault attack was applied to the *RSA* cryptosystem. Biham and Shamir proposed a new fault based attacking technique which is wildly known as Differential Fault Analysis (DFA)[10]. DFA attack is a very powerful attack model which can threaten a large class of ciphers. However, the actual attack procedure may vary from cipher to cipher, and one has to exploit the fault propagations suitably to

Algorithm 1: AES Key Scheduling Algorithm

```
Input: $K$ the initial key of length $N_k$ bytes
Output: $K^r$ the round key where $0 \leq r \leq N_r$
for $i = 0$ to $N_k - 1$ do
    $W[i] \leftarrow \{K[4*i], K[4*i+1], K[4*i+2], K[4*i+3]\}$;
for $i = N_k$ to $N_b * (N_r - 1)$ do
    $temp \leftarrow W[i-1]$;
    if $i \bmod N_k = 0$ then
        $temp \leftarrow SubWord(RotWord(temp)) \oplus Rcon[i/N_k]$;
    else if $N_k > 6$ and $i \bmod N_k = 4$ then
        $temp \leftarrow SubWord(temp)$;
    $W[i] \leftarrow W[i - N_k] \oplus temp$;
return $W$
```

extract the key of a given cipher. The foremost DFA proposed was on the DES cipher, which is essentially a Feistel cipher. Later, DFA has been extensively applied on other ciphers, with greater focus on the AES algorithm. Before discussing on fault based analysis of the AES cipher, we would discuss a general idea on DFA of block ciphers. We would restrict ourselves to the Substitution Permutation Network (SPN), as AES belongs to this family. However, similar observations and results can be obtained for Feistel structures, putting to threat all block ciphers of the modern day.

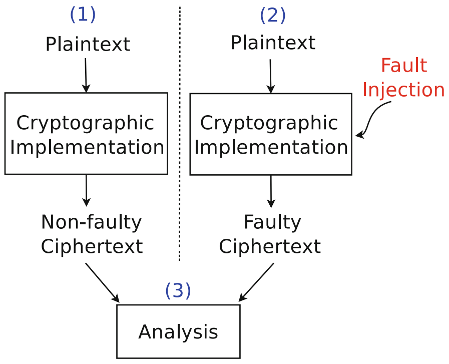
3.1 General Principle of DFA of Block Ciphers

In this section, we study the basic principle of DFA which shall be subsequently applied for the AES algorithm. As apparent from the name, DFA combines the concepts of differential cryptanalysis with that of fault attacks. DFA is applicable to almost any secret key cryptosystem proposed so far in the open literature such as DES, IDEA, and RC5 [10].

There has been considerable number of work about DFA of AES. Some of the DFA proposals are based on theoretical model [11, 15, 16, 36, 40, 45, 46, 54], while others launched successful attacks on ASIC and FPGA devices using previously proposed theoretical models [2, 8, 28, 46, 49]. The key idea of DFA is composed of three steps as shown in Fig. 3. (1) Run the cryptographic algorithm and obtain non-faulty ciphertexts. (2) Inject faults, i.e., unexpected environmental conditions into cryptographic implementations, rerun the algorithm with the same input, and obtain faulty ciphertexts (3) Analyze relationship between the non-faulty and faulty ciphertexts to significantly reduce the key space.

Practicality of DFA depends on the underlying fault model and the number of faulty ciphertext pairs needed. In the following section we will analyze all the fault models DFA of AES uses and point out their relationships. In this section, we continue the discussion on the working principle of DFA w.r.t. a generalized block cipher model.

Fig. 3 Three steps of DFA



DFA works under the assumption of underlying faults. These faults are often caused by various mechanisms, like: fluctuation of operating voltage, varying the clock frequency, changing the temperature of a device and with the most accurate injection of laser beams. However, in all of the above techniques the faults are created by sudden variation of the operating conditions. It may be noted that apart from the above means of malicious or intentional fault injections, faults can be also unintentional. With the growing complexity of crypto devices, chances of mistakes in the design also increase.

Faults can be categorized depending on whether they are permanent or transient. From the point of view of cryptography, we would like to point out that transient faults are of high concern as they are hard to detect. These faults can be of such a short duration that most simulation based techniques of fault detection may be unable to detect the advent of the faults. However, as we shall soon observe that few faults are enough to leak the entire key of a standard cipher, like AES.

3.1.1 Fault Models

The faults can be of varying nature but can be categorized as follows:

1. **Bit Model**: This fault model assumes that the faults is localized to one bit. The fault control is crucial here, as there is a high probability that a random fluctuation of the operating conditions can lead to more than one bit getting affected. Hence attacks based on such models are often unrealistic and may not be practically viable.
2. **Single Byte**: A more practical and most common fault model is the single byte model. This fault model assumes that the faults are spread to bytes and the fault model can be any *random* non-zero value. This non-specificity of the fault value makes these types of DFAs very powerful and practical techniques.
3. **Multiple Byte**: In this fault model, it is assumed that the faults propagate to more than 1 byte. More often, these models are more practical, in the sense that the DFAs based on them work even with lesser fault control. In context to DFA of

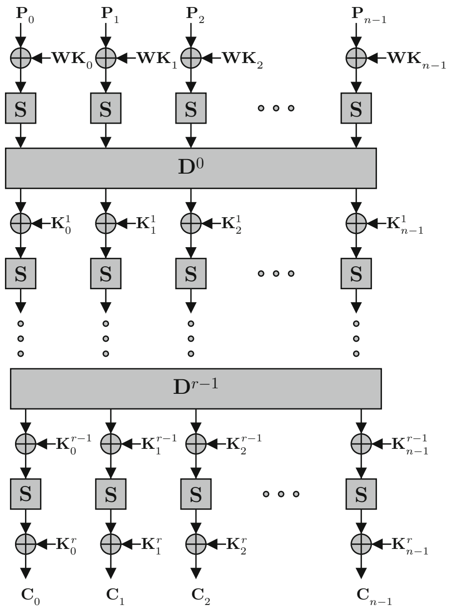


Fig. 4 Basic structure of SPN ciphers

AES, we shall observe a special fault model, namely the *Diagonal Fault Model* which helps to generalize the DFA of AES to a large extent. The fault values are again arbitrary, and hence makes these attacks very powerful.

3.1.2 The Effect of Faults on a Block Cipher

It is expected that the induced fault changes certain bits or bytes during a particular round of the encryption and generates certain differences. Most often the DFAs target the non-linear transformations, namely S-Boxes of the block ciphers. As the faults are induced during the encryption process, the fault propagation patterns give some *relations* between the input and output difference of certain S-boxes. In most of the ciphers like AES, the S-boxes are known and therefore, one can easily deduce the *difference distribution table* of the S-box being used. Generally, the S-boxes have inputs which are combined with part of the keys through some mixing operation. Using the difference distribution table and the relations between the input and output difference one reduces the search space of a part of the key. This divide and conquer mechanism helps to recover the entire key quite efficiently for most ciphers. We explain the working in more details for the generalized SPN cipher, as modeled in Fig. 4.

Figure 4 shows the basic structure of r-round Substitution Permutation Network (SPN) cipher with block length n-bytes. Each round consists of confusion layer S which is realized by non-linear S-box operation, and a linear transformation called diffusion layer D, followed by an addition with the round key. There is an addition with the whitening key *WK* at the beginning of the encryption called key-whitening phase. The diffusion layer is generally provided by multiplication with *MDS* matrix followed by some rotation operations. The diffusion operation plays a major role in DFA. If a byte is modified at the input of the diffusion operation, the induced difference spreads to multiple bytes at the output depending on the *branch number* of the diffusion layer. The disturbed bytes are often referred to as the active bytes in the literature of differential cryptanalysis. Branch number for a diffusion matrix on bytes is used to observe how a non-zero input differential spreads in a cipher through the diffusion layer. Branch number is defined as the sum of the minimum number of active bytes at the input and output of the diffusion layer.

As the diffusion layer is a linear operation w.r.t. the key mixing operation namely XOR, the output difference can be expressed as a linear relation of the input differences. The attacker exploits these properties in the following fashion to obtain the key. Say a single byte fault is induced at the input of $(r-1)$th (penultimate) round and the corresponding difference at the input of D^{r-1} is $\alpha \neq 0$. If the branch number of the diffusion layer is b, the input byte fault will spread to $b-1$ bytes $(\alpha_{\pi_0}, \ldots, \alpha_{\pi_{b-2}})$ at the output of D_{r-1}, where π denotes the transformation of the diffusion layer. Each of these active bytes then pass through the S-boxes, which non-linearly transform them. The attacker then represents these output bytes in terms of a pair of fault-free and faulty ciphertexts (C, C^*) as follows:

$$\alpha_{\pi_j} = S^{-1}(C_{\pi_j} \oplus K^r_{\pi_j}) \oplus S^{-1}(C^*_{\pi_j} \oplus K^r_{\pi_j}) \tag{6}$$

where $j \in \{0, \ldots, b-2\}$ and S^{-1} represent the inverse of the S-box operation. Now the attacker knows the S-box input difference $C_{\pi_j} \oplus C^*_{\pi_j}$. From the difference distribution table he knows on an average*few* values satisfy a chosen $(\alpha_{\pi_j}, C_{\pi_j} \oplus C^*_{\pi_j})$ pair.

Further, because of the linear mapping in D^{r-1}, α_{π_j} depends linearly on α. Therefore, the attacker guesses the value of α and get the values of α_{π_j} i.e. the output differences. Using the input–output difference he retrieves the value $C_{\pi_j} \oplus K_{\pi_j}$ from the difference distribution table of the S-box. As C_{π_j} and $C^*_{\pi_j}$ are known to the attacker, hence he can retrieve the value of K_{π_j}. The attacker may need to induce faults multiple times in order to get all the bytes of the round key.

In the next section, we present the fault models used for DFA of AES in the literature and a summary of all the attacks performed. Subsequently, we present the fault attacks on AES.

4 DFA and Associated Fault Models

DFA exploits a small subspace of all possible faults. The practicality of a fault attack largely relies on the underlying fault model: the nature of the faults and the ability to actually obtain such faults in practice. Any random fault is not attackable. Only certain fault models are feasible enough to reveal the secret key in practical time. In the following section, we classify the DFA fault models in four scenarios by the location and round in which faults are injected.

4.1 Fault Models for DFA of AES

Table 1 is a summary of the published DFA of AES. Faults can be injected either (I) in AddRoundKey in round 0, (II) between the output of seventh and the input of eighth round MixColumns, or (III) between the output of eighth and the input of ninth round MixColumns. In each scenario, we analyze the (A) fault models, (B) number of faulty ciphertexts needed, (C) the key space for brute force after obtaining the faulty outputs to recover the secret, and (D) the experimental validation of the attack. The considered transient faults are categorized into single bit, single byte, and multiple byte transient faults. It may be noted that we have purposefully omitted faults of permanent nature: namely stuck-at-1 or stuck-at-0 as they are not relevant from the DFA perspective. Rather transient faults are more relevant, because of their stealthy nature and ability to defeat counter-measures for classical fault tolerance. For detailed discussion in this direction, we would redirect the author to [56].

In the following discussions in this section we elaborate the fault models present in Table 1.

Table 1 A summary of DFA of AES

Fault model			No. of faulty CTs ⋆	Key space	Experiment
Section 4.1.1 Faults are injected in AddRoundKey in round 0					
Single bit		[11]	128	1	No
Section 4.1.2 Faults are injected between the output of seventh and the input of eighth round MixColumns					
Single byte		[45]	2	2^{40}	Underpowering [28, 49]
		[40]	2	2^{32}	No
		[54]	1	2^{8}	No
Multiple byte	DM0	[46]	1	2^{32}	Overclocking [46]
	DM1	[46]	1	2^{64}	
	DM2	[46]	1	2^{96}	
	DM3	[46]	2^{128}	2^{128}	
Section 4.1.3 Faults are injected between the output of eighth and the input of ninth round MixColumns					
Single bit		[16]	≈ 50	1	Overclocking [2]
Single byte		[15]	≈ 40	1	Underpowering [8]
		[36]†	6	1	No
Multiple byte	DM0	[36]‡	6	1	No
	DM0	[36]◇	1500	1	No

⋆ *CT* ciphertext, † Only 1 byte in a word is faulty, ‡ 2 or 3 bytes in a word are faulty, ◇ All 4 bytes in a word are faulty

4.1.1 Faults are Injected in AddRoundKey in Round 0

The only fault model an attacker uses in this scenario is single bit transient fault.

Single Bit Transient Fault In [11], the attacker is able to set or reset every bit of the first round key one bit at a time. This attack recovers the entire key using 128 faulty ciphertexts with each faulty ciphertext uniquely revealing one key bit. Hence, the key space required to reveal the key is one. However, as transistor size scales, this attack becomes impractical even with expensive equipments such as lasers to inject the faults, because it requires precise control of the fault location [1].

4.1.2 Faults are Injected Between the Output of Seventh and the Input of Eighth MixColumns

The attacker uses various fault models and analysis in this scenario including single and multiple byte fault.

Single Byte Transient Fault The three different attacks using this fault model are shown in Table 1. In the first DFA [45], two faulty ciphertexts are needed to obtain the key. This fault model is experimentally verified in [28, 49]. In [49], underpowering is used to inject faults into a smart card with AES ASIC implementation. Although no more than 16 % of the injected faults fall into the single byte fault category, only 13 faulty ciphertexts are needed to obtain the key. In [28], the authors underpower an AES FPGA implementation to inject faults with a probability of 40 % for single byte fault injection.

Multiple Byte Transient Fault Saha et al. [46] proposes a general byte fault model called **diagonal fault model**. The authors divide the AES state matrix into four different diagonals and each diagonal has 4 bytes. **A diagonal** is a set of 4 bytes of the AES state matrix, where the ith diagonal is defined as follows:

$$D_i = \{s_{j,(j+i)mod4}\ ;\ 0 \leq j < 4\} \tag{7}$$

We obtain the following four diagonals.

$$D_0 = (s_{0,0}, s_{1,1}, s_{2,2}, s_{3,3}),\ D_1 = (s_{0,1}, s_{1,2}, s_{2,3}, s_{3,0}),$$
$$D_2 = (s_{0,2}, s_{1,3}, s_{2,0}, s_{3,1}),\ D_3 = (s_{0,3}, s_{1,0}, s_{2,1}, s_{3,2})$$

Fault in diagonal D_i will affect the entire ith column of the state matrix after the `MixColumns` operation. The diagonal fault model is classified into four different cases, denoted as DM0, DM1, DM2, and DM3. As shown in Fig. 5, for DM0, faults can be injected in one of the diagonals; D_0, D_1, D_2, or D_3. For DM1, faults can be injected in at most two diagonals. For DM2, faults can be injected in at most three diagonals. Finally, for DM3, faults can be injected in at most four diagonals (Fig. 5).

The authors also validate the diagonal fault model with a practical fault attack on AES FPGA implementation using overclocking.

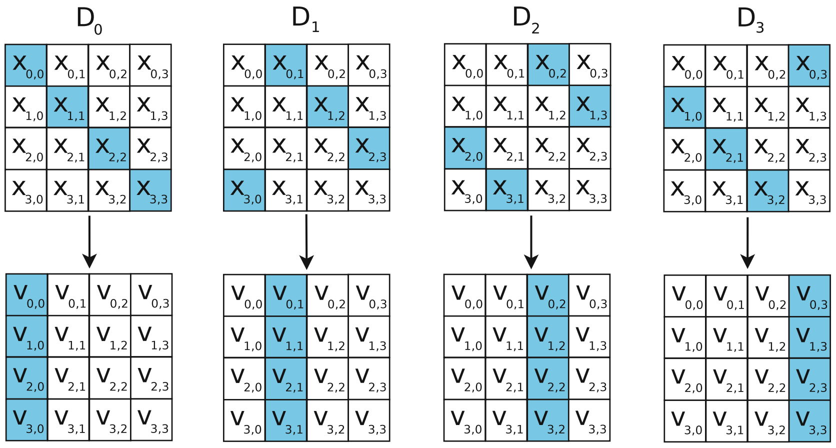

D_0
D_1
D_2
D_3
$X_{0,0}$ $X_{0,1}$ $X_{0,2}$ $X_{0,3}$
$X_{1,0}$ $X_{1,1}$ $X_{1,2}$ $X_{1,3}$
$X_{2,0}$ $X_{2,1}$ $X_{2,2}$ $X_{2,3}$
$X_{3,0}$ $X_{3,1}$ $X_{3,2}$ $X_{3,3}$
$V_{0,0}$ $V_{0,1}$ $V_{0,2}$ $V_{0,3}$
$V_{1,0}$ $V_{1,1}$ $V_{1,2}$ $V_{1,3}$
$V_{2,0}$ $V_{2,1}$ $V_{2,2}$ $V_{2,3}$
$V_{3,0}$ $V_{3,1}$ $V_{3,2}$ $V_{3,3}$

Fig. 5 Fault propagation of diagonal faults. The *upper row* shows the diagonals that faults are injected in. The *lower row* shows the corresponding columns being affected

4.1.3 Faults are Injected Between the Output of Eighth and the Input of Ninth MixColumns

Single Bit Transient Fault In [16], the attacker needs only three faulty ciphertexts to succeed with a probability of 97 %. The key space is trivial. Agoyan et al. [2] validates this single bit attack on a Xilinx 3AN FPGA using overclocking. It is reported that the success rate of injecting this kind of fault is 90 %.

Single Byte Transient Fault In [15], the authors use a byte level fault model. They are able to obtain the key with 40 faulty ciphertexts, and the key is uniquely revealed. This model is used in a successful attack by underpowering a 65 nm ASIC chip [8]. In this attack, 3,9881 faulty ciphertexts are collected during the ten experiments; 3,0386 of them were actually the outcome of a single byte fault. Thus, it has a successful injection rate of 76 %.

Multiple Byte Transient Fault Moradi et al. [36] presents a DFA of AES when the faults are injected in a 32-bit word. The authors propose two fault models. In the first model, they assume that at least one of the bytes among the four targeted bytes is non-faulty. This means the number of faulty bytes can be 1, 2, or 3 bytes. So this fault model includes the single byte fault model. If only one single byte fault is injected, 6 faulty ciphertexts are required to reveal the secret key. Whereas the second fault model requires around 1500 faulty ciphertexts. These faulty ciphertexts derive the entire key at constant time. Though the second fault model is much more general, the amount of faulty ciphertexts it requires is very large, it is difficult for the attacker to get all the ciphertexts without triggering the CED alarm.

In summary, the attacker can obtain the secret key with one or two faulty ciphertexts when single or multiple byte transient faults are injected. In the following subsection, we present a detailed analysis on the inter-relationships of the fault models discussed so far.

4.2 Relationships Between the Discussed Fault Models

As previously mentioned, DFA of AES does not exploit all possible faults. Rather, it exploits a subset of faults, namely single bit, single byte, and multiple byte transient faults injected in selected locations and rounds. Therefore, understanding the relationships among various fault models is the basis for understanding and comparing the various fault attacks on AES. Further the inter-relationships developed also help in analyzing the security of the counter-measured: both conventional as well as for designing new DFA-specific CED. Because DFA of AES targets the last few rounds,[1] we synthesize the relationships between different fault models based on the locations and rounds they are injected in.

4.2.1 Faults are Injected in AddRoundKey in Round 0

As we mentioned previously, this attack uses a very restricted fault model, and it is not practical. Thus, this fault model is also not useful for the attacker.

4.2.2 Faults are Injected Between the Output of Seventh and the Input of Eighth MixColumns

Figure 6a summarizes the relationships between the DFA-exploitable fault models by injecting faults in the output of seventh round MixColumns and the input of eighth round MixColumns.

Single byte faults are, in turn, a subset of the DM0 faults which, in turn, are a subset of the DM1 faults, and so on. The relationship is summarized in (8).

$$\text{Single Byte} \subset DM0 \subset DM1 \subset DM2 \subset DM3 \tag{8}$$

A more careful look reveals that 2 byte faults can be either DM0 or DM1 but not DM2. Similarly, 3 byte fault can not be DM3. The relationship between faulty bytes from 5 to 12 and diagonal fault models are summarized in Fig. 6a.

As shown in Fig. 6a, DM3 includes all possible byte transient faults. The attacks proposed in [46] show that DFA based on DM0, DM1, and DM2 leads to the successful retrieval of the key. Remember that DM3 faults are the universe of all possible transient faults injected in the selected AES round. These faults spread across all four diagonals of the AES state and hence, are not vulnerable to DFA as mentioned in Sect. 4.1.2. These fault models are multiple byte transient faults and thus, attacks based on these models are more feasible than those based on single byte transient faults, which are a subset of the model DM0. The more encompassing the fault model is, the more realistic the attacks based on it are.

[1] In general, the practical faults used in DFA target the seventh, eighth, and ninth rounds.

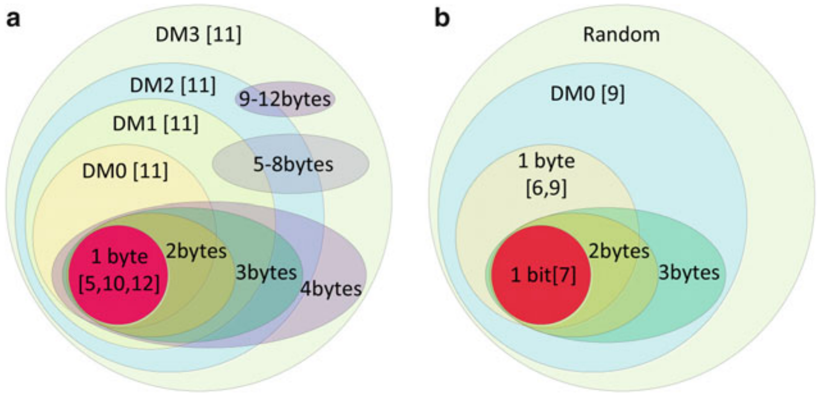


Fig. 6 Relationships between DFA fault models when faults are injected between (**a**) the output of seventh and the input of eighth round MixColumns, (**b**) output of eighth and the input of ninth round MixColumns

4.2.3 Faults are Injected Between the Output of Eighth and the Input of Ninth MixColumns

Figure 6b summarizes the relationships between the DFA-exploitable fault models by injecting faults in the output of eighth and the input of ninth round MixColumns. Single bit transient faults are a subset of single byte faults. Single byte faults are again a subset of DM0 faults. Two and three byte faults are a subset of DM0 faults. Again, attacks based on multiple byte faults are more feasible than those based on single bit and single byte faults.

In the following section, we detail the above mentioned fault attacks on AES.

5 Differential Fault Attacks on AES: Early Efforts

A very central property which is used in the algorithms to perform a DFA of AES is the differential features of the AES S-boxes. The following section presents the differential property of the AES S-Box.

5.1 Differential Properties of AES S-Box

In this section we discuss differential properties of S-box, which will be useful for DFA. In case of AES, the input to the S-box in each round is the XOR of previous round output and the round key. Figure 7a shows two S-box operations: one with normal input *in* and the other with a difference α to the input.

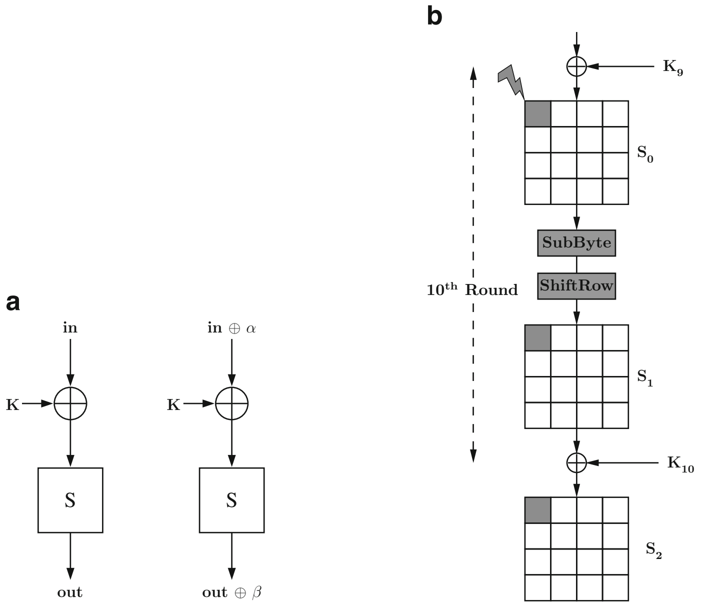


Fig. 7 Differential property of AES. (**a**) Difference across S-box. (**b**) Flow of fault in the last round

Here *in* is the previous round output byte and K is the round key byte. The AES S-box is a non-linear operation, therefore, input difference α will change to β at the S-box output *out*. Now if we replace the value of $in \oplus K$ by X, we can relate the input output differences by following equation:

$$\beta = S(X \oplus \alpha) \oplus S(X) \tag{9}$$

According to the properties of AES S-box for a particular value of α and β the above equation can have 0, 2, or 4 solutions of X [44]. For a fixed value of α, among the 256 possible values of β, only one value leads to four solutions of the equation and 126 values lead two solutions. The rest of the values will not produce any solution. This implies only 127 out of 256 choices of β produce solutions for X and the average number of solutions of X is one. It may also be noted that if we know the values of α, β, and *in*, we can get the values of K from the above equation. This property is being used in most of the advanced DFAs on AES. In the subsequent part of the chapter we explain DFA of AES using these properties.

5.2 DFA of AES Using Bit Faults

When AES was introduced at that time side-channel analysis and fault analysis were become two very prominent fields of research in the research community. The first DFA was already proposed on DES. Therefore, it was a challenge for the researchers in this field to analyze AES in the light of DFA. The initial attempts were further inspired by the fault injection techniques, which practically demonstrated that flipping a single bit of an intermediate computation result is possible using relatively less expensive devices like simple camera flash installed on a microscope or laser equipments [51]. However byte faults are more realistic compared to the bit faults.

In the following sections we discuss about DFA using bit level and byte level fault models. The induced fault is assumed to be random in nature and the attack algorithm is oblivious of the fault value. Let us first study the DFA which targets the last round of the AES encryption.

5.3 Bit Level DFA of Last Round of AES

In this attack, originally proposed in [16], it is assumed that the fault is induced at any particular bit of the last round input. However, the exact fault location, i.e., the exact bit where the fault is created is unknown.

Let us consider AES with 128-bit key for the sake of simplicity, though the discussion can be easily extended to the other AES versions. Figure 7b shows the flow of fault corresponding to the single-bit difference at the input of the tenth round. In the figure, K^9 and K^{10} denotes the ninth and tenth round keys respectively. The state matrices S_0, S_1 and S_2 show the corresponding XOR differences of the fault-free and the faulty states. As the fault is induced in a bit, therefore, the disturbance is confined within a single byte. So, from the XOR difference of fault-free and faulty ciphertexts one can easily get the location of the faulty byte (note that the bit is not evident because of the S-Box).

Consider the fault induced at the (i,j)th byte of the tenth round input state matrix S_0. Let x be the fault-free value at the tenth round input and ε be the corresponding fault value. Note that the attacker is aware of the fault-free (C) and faulty (C^*) ciphertexts. As already stated, from the XOR difference of C and C^* one can get the byte position (i,j), where the fault is induced. The (i,j) byte where the bit fault is induced in the byte x can be represented in terms of (C, C^*) as follows:

$$C_{i,l} \oplus C^*_{i,l} = SR(S(x_{i,j})) \oplus SR(S(x_{i,j} \oplus \varepsilon)) \tag{10}$$

Note that $l = (j - i) \bmod 4$ provides the corresponding column index, where the faulty byte in the jth column and ith row shifts due to the *Shiftrows* operation. In other words, the fault location (i,j) in the difference matrix S_0 changes to (i, l) at the tenth round output.

Having obtained the location of the fault, the attacker is now set to ascertain the value of the fault. Note the similarity of the above equation with that of Eq. (9). The value of $C_{i,l} \oplus C^*_{i,l}$ being known to the attacker, in order to get the value of $x_{i,j}$ he guesses eight possible values of ε. For each possible value of ε, he gets on an average one hypotheses for $x_{i,j}$, which will satisfy the above equation (refer Sect. 5.1). Thus, for all the eight possible values of ε, he gets on an average eight candidates for $x_{i,j}$. In order to identify the unique value for $x_{i,j}$ he obtains another faulty ciphertext by injecting another fault (i.e., the fault location is a different bit) in the same byte. A similar approach leads to another set of eight values for $x_{i,j}$. Intersection of these two sets from the two different faulty ciphertexts is expected to determine the exact value of $x_{i,j}$.

This same technique is repeated for the other bytes in order to get all the 16 bytes of x. On an average thus $2 \cdot 16 = 32$ faulty ciphertexts are needed to determine the value of the state matrix x. Thus, the attacker obtains the fault-free input of tenth round. Being aware of the fault-free ciphertext C, one can easily retrieve the tenth round key K^{10} from the relation $C = SR(S(x) \oplus K^{10})$. Then, as the AES key-schedule is invertible one trivially retrieve the master key.

5.4 Bit Level DFA of First Round of AES

In this section we describe another bit level DFA, where the fault is induced in the first round of AES encryption. This is a more general attack and applicable to most of the ciphers. The main difference of this attack from the previous ones and the others which follow is the underlying fault model. The fault model is a *bit reset* model, which implies that the attacker has capability to reset a specific bit at a targetted byte location of the AES encryption. The attacker targets the first key whitening operation before the *SubBytes* operation. The plaintext is set to a zero string of length 128 bits, denoted as P_{zero}. The plaintext is fixed throughout the attack and the objective of the attack is to obtain the whitening key K.

To start with, an encryption is done using P_{zero} and K under normal environment and the fault-free ciphertext C_{zero} is obtained and stored. Now a fault is induced according to the fault model discussed. It is assumed that the induced fault resets the lth bit of the (i, j) byte at the input to the first *SubBytes* operation. Let us assume that the fault free input to the *SubBytes* operation is x. Therefore, we can write $x = P_{zero} \oplus K$. The attacker tries to detect the value of l by repeating the following simple steps: He compares the fault-free ciphertext, C_{zero} with that of the faulty one, C^*_{zero}. If they are equal it implies that the lth bit of the (i, j)th byte of x, which was reset due to the fault, was already zero and thus the effect of the reset fault was inconsequential. Thus the corresponding bit of the (i, j)th key byte was zero (as the plaintext is all zero). On the other hand, a different value of C_{zero} and C^*_{zero} implies that the induced fault reset the bit $x^l_{i,j}$ with effect. That means the fault-free value of $x^l_{i,j}$ was one and after fault induction it changes to zero. This also means the corresponding bit value of K is one. The fault thus reveals whether a particular bit of the whitening key is one or zero. The same technique is repeated for all the 128 bits, and thus 128 faulty ciphertexts are needed to get the master key.

The attack is relatively simple in nature, however is relatively less practical. The assumed fault model is impractical as it requires very precise control over the fault location to enable fault in every bit positions. Thus fault attacks on AES with more relaxed fault models and lesser fault induction requirements are desirous and topics of the future sections.

6 State-of-the-Art DFAs on AES

In this section, we present an overview on some of the more recent fault attacks on AES. These attacks use more practical fault model, namely the byte faults.

6.1 Byte Level DFA of Penultimate Round of AES

In byte level DFA, we assume that certain bits of a byte is corrupted by the induced fault and the induced difference is confined within a byte. Due to the fact that the fault is induced in the penultimate round, implies that apart from using the differential properties of S-box (as used in the bit level DFA on last round of AES), the attacker also uses the differential properties of the MixColumns operation of AES. As already mentioned in the AES, diffusion is provided using a 4×4 MDS matrix in the MixColumns. Due to this matrix multiplication, if 1 byte difference is induced at the input of a round function, the difference is spread to 4 bytes at the round output.

Figure 8a shows the flow of fault. The induced fault has generated a single byte difference at the input of the ninth round MixColumns. Let f be the byte value of the difference and the corresponding 4-byte output difference is $(2f, f, f, 3f)$, where $2, 1$, and 3 are the elements of the first row of the MixColumns matrix. The 4-byte difference is again converted to (f_0, f_1, f_2, f_3) by the non-linear S-box operation in the tenth round. The ShiftRows operation will shift the differences to four different locations. The attacker has access to the fault-free ciphertext C and faulty ciphertext C^*, which differs only in 4 bytes. Now, we can represent the 4-byte difference $(2f, f, f, 3f)$ in terms of the tenth round key K^{10} and the fault-free and faulty ciphertexts by the following equations:

$$\begin{aligned}
2f &= S^{-1}(C_{0,0} \oplus K^{10}_{0,0}) \oplus S^{-1}(C^*_{0,0} \oplus K^{10}_{0,0}) \\
f &= S^{-1}(C_{1,3} \oplus K^{10}_{1,3}) \oplus S^{-1}(C^*_{1,3} \oplus K^{10}_{1,3}) \\
f &= S^{-1}(C_{2,2} \oplus K^{10}_{2,2}) \oplus S^{-1}(C^*_{2,2} \oplus K^{10}_{2,2}) \\
3f &= S^{-1}(C_{3,1} \oplus K^{10}_{3,1}) \oplus S^{-1}(C^*_{3,1} \oplus K^{10}_{3,1})
\end{aligned} \tag{11}$$

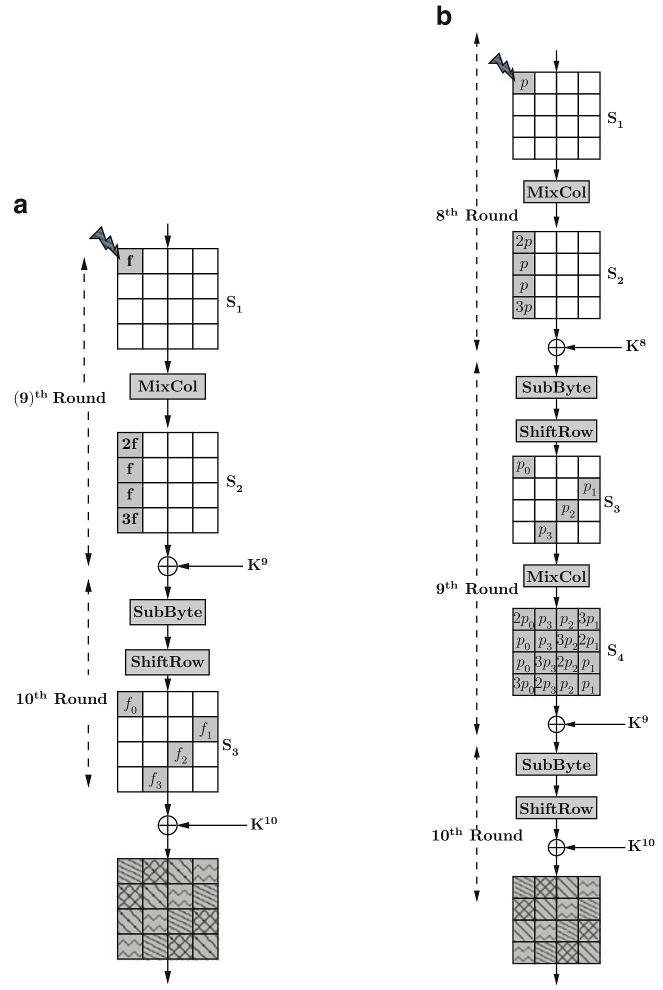


Fig. 8 Flow of faults in AES rounds. (**a**) Differences across the last two rounds. (**b**) Differences across the last three rounds

The above four equations can be expressed as the basic equation (9). Therefore, it can be represented in the form $A = B \oplus C$ where A, B, and C are bytes in $\mathbf{F}_{2^8}$, having 2^8 possible values each. Now a uniformly random choice of (A, B, C) is expected to satisfy the equation with probability $\frac{1}{2^8}$. Therefore, in this case 2^{16} out of 2^{24} random choices of (A, B, C) will satisfy the equation.

This fact can be generalized. Consider we have M such related equations. These M equations consist of N uniformly random byte variables. The probability that a random choice of N variables satisfy all the M equations simultaneously is $(\frac{1}{2^8})^M$. Therefore the reduced search space is given by $(\frac{1}{2^8})^M \cdot (2^8)^N = (2^8)^{N-M}$. For our case we have four equations which consist of five unknown variables: f, $K^{10}_{0,0}$, $K^{10}_{1,3}$, $K^{10}_{2,2}$, and $K^{10}_{3,2}$. Therefore, the four equations will reduce the search space of the variables to $(2^8)^{5-4} = 2^8$. That means out of 2^{32} hypotheses of the four key bytes, only 2^8 hypotheses will satisfy the above four equations. Therefore, using one fault the attacker can reduce the search space of the four key byte to 2^8. Using two such faulty ciphertexts one can uniquely determine the key quartet. This implies, for one key quartet, one has to induce two faults in the required location. For all the four key quartets i.e., the entire AES key, an attacker needs to induce eight faults. Therefore using eight faulty ciphertexts and a fault-free ciphertext, it is expected to uniquely determine the 128-bit key of AES.

6.1.1 DFA Using Two Faults

The attack can further be improved. It was shown in [45] that instead of inducing fault in ninth round, if we induce fault in between seventh and eighth round MixColumns, we can determine the 128-bit key using only two faulty ciphertexts. Figure 8b shows the spreading of faults when it is induced in such a fashion. The single byte difference at the input of eighth round MixColumns is spread to 4 bytes. The Shiftrows operation ensures that there is one disturbed byte in each column of the state matrix. Each of the 4-byte difference again spreads to 4 bytes at ninth round MixColumns output. Therefore the relation between the fault values in the four columns of difference state matrix S_4 is equivalent to four faults at four different columns of ninth round input state matrix as explained in the previous attack. This implies that using two such faults we can uniquely determine the entire AES key.

Note that the exact working of the DFA proposed in [45] is slightly different from above, though the underlying principle is the same. The attack maintains a list $\mathcal{D}$ for each column of the difference matrix S_4 assuming a 1-byte fault in the input of the penultimate round MixColumns. The size of the table $\mathcal{D}$ is thus 4×255 4-byte values, as the input fault can occur in any byte of a column and can take 255 non-zero values. Assuming that the fault occurs in the difference matrix S_3 in the first column, then equations similar to Eq. (11) can be written, with the left hand side of the equations being a 4-byte tuple $(\Delta_0, \Delta_1, \Delta_2, \Delta_3)$. It is expected that the correct guess of the keys $K^{10}_{0,0}, K^{10}_{1,3}, K^{10}_{2,2}$, and $K^{10}_{3,2}$ should provide a 4-byte tuple which belongs to the list $\mathcal{D}$. There are other wrong keys which also pass this test, and analysis shows that on an average 1036 elements pass this test with a single fault. Repeating the same for all the 4-columns of the difference matrix S_4, reduces the AES key to $1036^4 \approx 2^{40}$ (note that as the fault is assumed to be between seventh and eighth round each column of S_3 has a byte disturbed). However, if two faults are induced, the unique AES key is returned with a probability of 0.98.

This is the best known DFA of AES till date when the attacker does not have access to the plaintext and he needs to determine the key uniquely. However when the attacker has access to the plaintexts, he can still improve the attack by performing the DFA using only one fault and a further reduced brute force guess. Also it is possible to reduce the time complexity of the attack further from 2^{32} to 2^{30}.

6.1.2 DFA Using Only One Fault

The attack proposed in [45] can be further improved when the attacker has access to the plaintexts in addition to the ciphertexts [39]. In that case, he can do brute-force on the possible keys. The objective of this attack or its extensions is to perform the attack using only one fault. While a unique key may not be obtainable with a single fault, the AES key size can reduce to such a small size that a brute force search can be easily performed. It may be noted that reducing the number of fault requirements from 2 to 1 should not be seen in terms of its absolute values. In an actual fault attack, it is very unlikely that the attacker can have absolute control over the fault injection method and hence may need more number of trials. Rather, these attacks are capable of reducing the number of fault requirements by half compared to the attacks proposed in [45].

Consider Fig. 8b, where from the first column of S_4 we get four differential equations [similar to Eq. (11)] corresponding to the 4-tuple $(2p_0,\ p_0,\ p_0,\ 3p_0)$. Using these four differential equations we only guess the 2^8 values of p_0 and get the corresponding possible 2^8 hypotheses of the key quartet by applying the S-box difference distribution table. Therefore, one column of S_4 will reduce the search space of one quartet of key to 2^8 choices. Similarly, solving the differential equations from all the four columns we can reduce the search space of all the four key quartets to 2^8 values each. Hence, if we combine all the four quartets we get $(2^8)^4 = 2^{32}$ possible hypotheses of the final round key K^{10}. We have assumed here that the initial fault value was in the $(0, 0)$th byte of S_1. If we allow the fault to be in any of the 16 locations, the key space of AES is around 2^{36} values. This space can be brute force searched within 1 min and hence, shows that effectively one fault is sufficient to break AES.

The search space of the final round key can be further reduced if we consider the relation between the fault values at the state matrix S_2, which was not utilized in the previous attacks. This step serves as a second stage, which is coupled with the first stage on all the 2^{32} keys (for an assumed location of the faulty byte). We can represent the fault value in the first column of S_2 in terms of the ninth round key K^9 and the ninth round fault-free and faulty output C^9 and C^{*9} respectively by the following four differential equations:

$$\begin{aligned}2p = {} & S^{-1}(14(C^9_{0,0} \oplus K^9_{0,0}) \oplus 11(C^9_{1,0} \oplus K^9_{1,0}) \oplus \\ & 13(C^9_{2,0} \oplus K^9_{2,0}) \oplus 9(C^9_{3,0} \oplus K^9_{3,0})) \oplus \\ & S^{-1}(14(C^{*9}_{0,0} \oplus K^9_{0,0}) \oplus 11(C^{*9}_{1,0} \oplus K^9_{1,0}) \oplus \\ & 13(C^{*9}_{2,0} \oplus K^9_{2,0}) \oplus 9(C^{*9}_{3,0} \oplus K^9_{3,0}))\end{aligned} \tag{12a}$$

$$\begin{aligned}p = {} & S^{-1}(9(C^9_{0,3} \oplus K^9_{0,3}) \oplus 14(C^9_{1,3} \oplus K^9_{1,3}) \oplus \\ & 11(C^9_{2,3} \oplus K^9_{2,3}) \oplus 13(C^9_{3,3} \oplus K^9_{3,3})) \oplus \\ & S^{-1}(9(C^{*9}_{0,3} \oplus K^9_{0,3}) \oplus 14(C^9_{1,3} \oplus K^{*9}_{1,3}) \oplus \\ & 11(C^{*9}_{2,3} \oplus K^9_{2,3}) \oplus 13(C^{*9}_{3,3} \oplus K^9_{3,3}))\end{aligned} \tag{12b}$$

$$\begin{aligned}p = S^{-1}(13(C^9_{0,2} \oplus K^9_{0,2}) \oplus 9(C^9_{1,2} \oplus K^9_{1,2})\oplus\\ 14(C^9_{2,2} \oplus K^9_{2,2}) \oplus 11(C^9_{3,2} \oplus K^9_{3,2}))\oplus\\ S^{-1}(13(C^{*9}_{0,2} \oplus K^9_{0,2}) \oplus 9(C^{*9}_{1,2} \oplus K^9_{1,2})\oplus\\ 14(C^{*9}_{2,2} \oplus K^9_{2,2}) \oplus 11(C^{*9}_{3,2} \oplus K^9_{3,2}))\end{aligned} \tag{12c}$$

$$\begin{aligned}3p = S^{-1}(11(C^9_{0,1} \oplus K^9_{0,1}) \oplus 13(C^9_{1,1} \oplus K^9_{1,1})\oplus\\ 14(C^9_{2,1} \oplus K^9_{2,1}) \oplus 9(C^9_{3,1} \oplus K^9_{3,1}))\oplus\\ S^{-1}(11(C^{*9}_{0,1} \oplus K^9_{0,1}) \oplus 13(C^{*9}_{1,1} \oplus K^9_{1,1})\oplus\\ 14(C^{*9}_{2,1} \oplus K^9_{2,1}) \oplus 9(C^{*9}_{3,1} \oplus K^9_{3,1}))\end{aligned} \tag{12d}$$

In order to utilize the above equations we need the ninth round key. The ninth round key can be derived from the final round key by the following conversion matrix:

$$\begin{pmatrix} (K^{10}_{0,0} \oplus S[K^{10}_{1,3} \oplus K^{10}_{1,2}] \oplus h_{10}) & K^{10}_{0,1} \oplus K^{10}_{0,0} & K^{10}_{0,2} \oplus K^{10}_{0,1} & K^{10}_{0,3} \oplus K^{10}_{0,2} \\ (K^{10}_{1,0} \oplus S[K^{10}_{2,3} \oplus K^{10}_{2,2}]) & K^{10}_{1,1} \oplus K^{10}_{1,0} & K^{10}_{1,2} \oplus K^{10}_{1,1} & K^{10}_{1,3} \oplus K^{10}_{1,2} \\ (K^{10}_{2,0} \oplus S[K^{10}_{3,3} \oplus K^{10}_{3,2}]) & K^{10}_{2,1} \oplus K^{10}_{2,0} & K^{10}_{2,2} \oplus K^{10}_{2,1} & K^{10}_{2,3} \oplus K^{10}_{2,2} \\ (K^{10}_{3,0} \oplus S[K^{10}_{0,3} \oplus K^{10}_{0,2}]) & K^{10}_{3,1} \oplus K^{10}_{3,0} & K^{10}_{3,2} \oplus K^{10}_{3,1} & K^{10}_{3,3} \oplus K^{10}_{3,2} \end{pmatrix}.$$

Thus for each of the possible hypotheses of K^{10} produced by the first stage, and using the ciphertexts, (C, C^*), we get the values of (K^9, C^9, C^{*9}). Then the attacker tests the above four equations with these values. If satisfies, the candidate key is accepted, else rejected. For completeness, we state the detailed equations as follows:

$$\begin{aligned}2p = S^{-1}\big[14(S^{-1}[K^{10}_{0,0} \oplus C_{0,0}] \oplus K^{10}_{0,0} \oplus S[K^{10}_{1,3} \oplus K^{10}_{1,2}] \oplus h_{10})\oplus\\ 11(S^{-1}[K^{10}_{1,3} \oplus C_{1,3}] \oplus K^{10}_{1,0} \oplus S[K^{10}_{2,3} \oplus K^{10}_{2,2}])\oplus\\ 13(S^{-1}[K^{10}_{2,2} \oplus C_{2,2}] \oplus K^{10}_{2,0} \oplus S[K^{10}_{3,3} \oplus K^{10}_{3,2}])\oplus\\ 9(S^{-1}[K^{10}_{3,1} \oplus C_{3,1}] \oplus K^{10}_{3,0} \oplus S[K^{10}_{0,3} \oplus K^{10}_{0,2}])\big]\oplus\\ S^{-1}\big[14(S^{-1}[K^{10}_{0,0} \oplus C^*_{0,0}] \oplus K^{10}_{0,0} \oplus S[K^{10}_{1,3} \oplus K^{10}_{1,2}])\oplus\\ 11(S^{-1}[K^{10}_{1,3} \oplus C^*_{1,3}] \oplus K^{10}_{1,0} \oplus S[K^{10}_{2,3} \oplus K^{10}_{2,2}])\oplus\\ 13(S^{-1}[K^{10}_{2,2} \oplus C^*_{2,2}] \oplus K^{10}_{2,0} \oplus S[K^{10}_{3,3} \oplus K^{10}_{3,2}])\oplus\\ 9(S^{-1}[K^{10}_{3,1} \oplus C^*_{3,1}] \oplus K^{10}_{3,0} \oplus S[K^{10}_{0,3} \oplus K^{10}_{0,2}])\big]\end{aligned} \tag{13}$$

Similarly, the other three faulty bytes can be expressed by the following equations:

$$\begin{aligned}
p = S^{-1}\big[&9(S^{-1}[K_{0,3}^{10} \oplus C_{0,3}] \oplus K_{0,3}^{10} \oplus K_{0,2}^{10})\oplus\\
&14(S^{-1}[K_{1,3}^{10} \oplus C_{1,3}] \oplus K_{1,3}^{10} \oplus K_{1,2}^{10})\oplus\\
&11(S^{-1}[K_{2,1}^{10} \oplus C_{2,1}] \oplus K_{2,3}^{10} \oplus K_{2,2}^{10})\oplus\\
&13(S^{-1}[K_{3,0}^{10} \oplus C_{3,0}] \oplus K_{3,3}^{10} \oplus K_{3,2}^{10})\big]\oplus\\
S^{-1}\big[&9(S^{-1}[K_{0,3}^{10} \oplus C_{0,3}] \oplus K_{0,3}^{10} \oplus K_{0,2}^{10})\oplus\\
&14(S^{-1}[K_{1,3}^{10} \oplus C_{1,3}] \oplus K_{1,3}^{10} \oplus K_{1,2}^{10})\oplus\\
&11(S^{-1}[K_{2,1}^{10} \oplus C_{2,1}] \oplus K_{2,3}^{10} \oplus K_{2,2}^{10})\oplus\\
&13(S^{-1}[K_{3,0}^{10} \oplus C_{3,0}] \oplus K_{3,3}^{10} \oplus K_{3,2}^{10})\big]\oplus
\end{aligned} \tag{14}$$

$$\begin{aligned}
p = S^{-1}\big[&13(S^{-1}[K_{0,2}^{10} \oplus C_{0,2}] \oplus K_{0,2}^{10} \oplus K_{0,1}^{10})\oplus\\
&9(S^{-1}[K_{1,1}^{10} \oplus C_{1,1}] \oplus K_{1,2}^{10} \oplus K_{1,1}^{10})\oplus\\
&14(S^{-1}[K_{2,0}^{10} \oplus C_{2,0}] \oplus K_{2,2}^{10} \oplus K_{2,1}^{10})\oplus\\
&11(S^{-1}[K_{3,3}^{10} \oplus C_{3,3}] \oplus K_{3,2}^{10} \oplus K_{3,1}^{10})\big]\oplus\\
S^{-1}\big[&13(S^{-1}[K_{0,2}^{10} \oplus C^{*}_{0,2}] \oplus K_{0,2}^{10} \oplus K_{0,1}^{10})\oplus\\
&9(S^{-1}[K_{1,1}^{10} \oplus C^{*}_{1,1}] \oplus K_{1,2}^{10} \oplus K_{1,1}^{10})\oplus\\
&14(S^{-1}[K_{2,0}^{10} \oplus C^{*}_{2,0}] \oplus K_{2,2}^{10} \oplus K_{2,1}^{10})\oplus\\
&11(S^{-1}[K_{3,3}^{10} \oplus C^{*}_{3,3}] \oplus K_{3,2}^{10} \oplus K_{3,1}^{10})\big]
\end{aligned} \tag{15}$$

$$\begin{aligned}
3p = S^{-1}\big[&11(S^{-1}[K_{0,1}^{10} \oplus C_{0,1}] \oplus K_{0,1}^{10} \oplus K_{0,0}^{10})\oplus\\
&13(S^{-1}[K_{1,0}^{10} \oplus C_{1,0}] \oplus K_{1,1}^{10} \oplus K_{1,0}^{10})\oplus\\
&14(S^{-1}[K_{2,3}^{10} \oplus C_{2,3}] \oplus K_{2,1}^{10} \oplus K_{2,0}^{10})\oplus\\
&9(S^{-1}[K_{3,2}^{10} \oplus C_{3,2}] \oplus K_{3,1}^{10} \oplus K_{3,0}^{10})\big]\oplus\\
S^{-1}\big[&11(S^{-1}[K_{0,1}^{10} \oplus C^{*}_{0,1}] \oplus K_{0,1}^{10} \oplus K_{0,0}^{10})\oplus\\
&13(S^{-1}[K_{1,0}^{10} \oplus C^{*}_{1,0}] \oplus K_{1,1}^{10} \oplus K_{1,0}^{10})\oplus\\
&14(S^{-1}[K_{2,3}^{10} \oplus C^{*}_{2,3}] \oplus K_{2,1}^{10} \oplus K_{2,0}^{10})\oplus\\
&9(S^{-1}[K_{3,2}^{10} \oplus C^{*}_{3,2}] \oplus K_{3,1}^{10} \oplus K_{3,0}^{10})\big]
\end{aligned} \tag{16}$$

We thus get four differential equations and the combined search space of (K^9, C^9, C^{*9}) and p is $2^{32} \cdot 2^8 = 2^{40}$. Therefore, the four equations will reduce this search space of K_{10} to $\frac{2^{40}}{(2^8)^4} = 2^8$. Hence, using only one faulty ciphertext, one can reduce the search space of AES-128 key to 256 choices. However, the time complexity of the attack is 2^{32}, as we have to test all the hypothesis of K^{10} by the above four equations. In the next subsection, we present an improvement to reduce the time complexity of the attack to 2^{30} from 2^{32}.

6.1.3 DFA with Reduced Time Complexity

The above second phase of the analysis is based on four equations: (13), (14), (15), and (16). All the 2^{32} possible key hypotheses are tested by these four equations. The key hypotheses which are satisfied by all four equations are considered and rest are discarded.

However, if we consider the above four equations in pairs we observe that each possible pair does not contain all the 16 bytes of the AES key. For example, the pair of Eqs. (14) and (15) contains 14 key bytes excluding $K^{10}_{0,0}$ and $K^{10}_{0,1}$. This fact can be utilized to reduce the time complexity of the attack. We use this observation to split the lists of key which are exported in the first phase of the attack and subsequently filtered in the second phase.

In the first phase of the attack we have four quartets $(K^{10}_{0,0}, K^{10}_{1,3}, K^{10}_{2,2}, K^{10}_{3,1})$, $(K^{10}_{0,1}, K^{10}_{1,0}, K^{10}_{2,3}, K^{10}_{3,2})$, $(K^{10}_{0,2}, K^{10}_{1,1}, K^{10}_{2,0}, K^{10}_{3,3})$, and $(K^{10}_{0,3}, K^{10}_{1,2}, K^{10}_{2,1}, K^{10}_{3,0})$ Let us assume one value of the first quartet is (a_1, b_1, c_1, d_1). As per the property of the S-Box, there will be another value a_2 of $K^{10}_{0,0}$, which satisfies the system of equations generated from the first column of S_4, with the rest of the key byte values remaining same.

Using this idea, we can divide the list for the quartet $(K^{10}_{0,0}, K^{10}_{1,3}, K^{10}_{2,2}, K^{10}_{3,1})$ into two sublists, L_1, L_2. As depicted in Fig. 9 The list L_1 contains the pair values for the key byte $K^{10}_{0,0}$ (note that the key byte $K^{10}_{0,0}$ has always an even number of possible choices). The list L_2 contains the distinct values for the remaining part of the quartet, $(K^{10}_{1,3}, K^{10}_{2,2}, K^{10}_{3,1})$. Thus the expected size of the lists L_1 and L_2 is 2^7 each, compared to the previous list size of 2^8 with 4-tuple $(K^{10}_{0,0}, K^{10}_{1,3}, K^{10}_{2,2}, K^{10}_{3,1})$.

Similarly, we store the possible values of quartet $(K^{10}_{0,1}, K^{10}_{1,0}, K^{10}_{2,3}, K^{10}_{3,2})$ in two lists, L_3 and L_4. Here L_3 stores the pair values for the key byte $K^{10}_{0,1}$, while the list L_4 contains the distinct values for the key bytes $(K^{10}_{1,0}, K^{10}_{2,3}, K^{10}_{3,2})$. Next, we select the key bytes from the six lists, $L_1, L_2, L_3, L_4, L_5, L_6$, to solve the equations of the second phase of the attack such that the time complexity is reduced.

Because of the observations regarding the pair of Eqs. (13), (16) and (14), (15), the second phase can be divided into two parts. In part one, we test the keys generated from the first phase of the attack by the pair of Eqs. (14) and (15). In Fig. 9 this is denoted as *Test1*. As the two equations for *Test1* does not require key bytes $K^{10}_{0,0}$ and $K^{10}_{0,1}$, we only consider all possible keys generated from lists L_2, L_4, L_5, L_6. There are 2^{30} such possible keys. In the second part we combine each of the 14 byte keys satisfying *Test1* with one of the four possible values arising out of the four combinations of the pair of values for $K^{10}_{0,0}$ in L_1 and $K^{10}_{0,1}$ in L_3. These keys are further tested in parallel by Eqs. (13) and (16). In Fig. 9, we refer to this test as *Test2*.

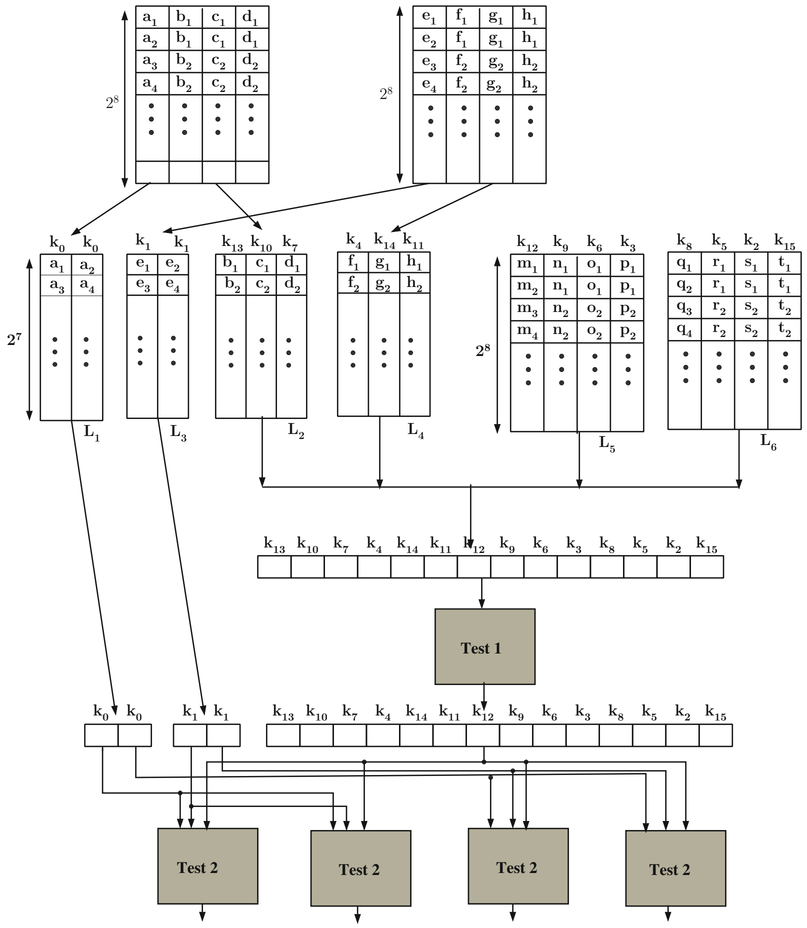


Fig. 9 Model for data-flow parallelization in the second phase

The size of the lists L_2 and L_4 is 2^7; and the size of lists L_5 and L_6 is 2^8. Therefore the number of possible keys generated from this four lists is $2^7 \times 2^7 \times 2^8 \times 2^8 = 2^{30}$. These 2^{30} keys are fed as input to *Test1* which is expected to reduce the key hypotheses by 2^8. Therefore each instance of *Test2* will receive input of $(\frac{2^{30}}{2^8}) = 2^{22}$ expected key hypotheses. The chance of each key satisfying *Test2* is 2^{-16} which implies each instance of *Test2* will result in 2^6 key hypotheses.

It may be easily observed that the time required is because of step 3, which is equal to 2^{30}, making the overall attack four times faster on an average, and still reducing the overall keyspace of AES to around 2^8 values.

The above fault models are based on single byte fault models, which assume that the fault is localized in a single byte. However due to impreciseness in the fault induction, the fault can spread to more than 1 bytes. Such a *multiple-byte* fault requires a revisit at the DFA methods. In [47], a technique for performing DFA when such faults occur where presented, which generalize further the DFA proposed in [6] and later extended in [30]. The underlying fault models assumed in this attack were already introduced in Sect. 4.1.3 and were called as *diagonal fault models*. In the next section, we outline the idea of these attacks.

7 Multiple Byte DFA of AES-128

In this section, we present the DFAs under the multiple byte fault models. The DFAs are efficient to obtain the AES key using 2–4 faults, when the faults corrupt upto three diagonals of the four diagonals of the AES state matrix at the input of the eighth round MixColumns. In the next subsection, we first observe the DFAs when the fault is confined to one diagonal of the state matrix, i.e., the fault is according to the fault model $DM0$.

7.1 DFA According to Fault Model $DM0$

We first show that faults which are confined to one diagonal are equivalent and can be used to retrieve the key using the same method.

7.1.1 Equivalence of Faults in the Same Diagonal

Let us first observe the propagation of a fault injected in diagonal D_0 through the round transformations from the input of the eighth round to the output of the ninth round.

Figure 10 shows some cases of fault induction in diagonal D_0. The faults vary in the number of bytes that are faulty in D_0 at the input of the 8 round. We emphasize the fact that irrespective of the number or positions of bytes that are faulty in D_0, due to the subsequent ShiftRows operation the fault is confined to the first column C_0 of the state matrix at the end of the eighth round. So the fault propagation in the ninth round for all these cases is similar and leads to the same byte inter-relations at the end of the ninth round.

In general any fault at the input of the eighth round in the ith diagonal, $0 \leq i \leq 3$, leads to the ith column being affected at the end of the round. There are four diagonals and faults in each diagonal maps to four different byte inter-relations at the end of the ninth round. These relations are depicted in Fig. 11. These relations will remain unchanged for any combination of faults injected within a particular diagonal. Each of the four sets of relations in Fig. 11 will be used to form key dependent equations. Each of the equation sets will comprise of four equations of similar nature as shown in Eq. (11).

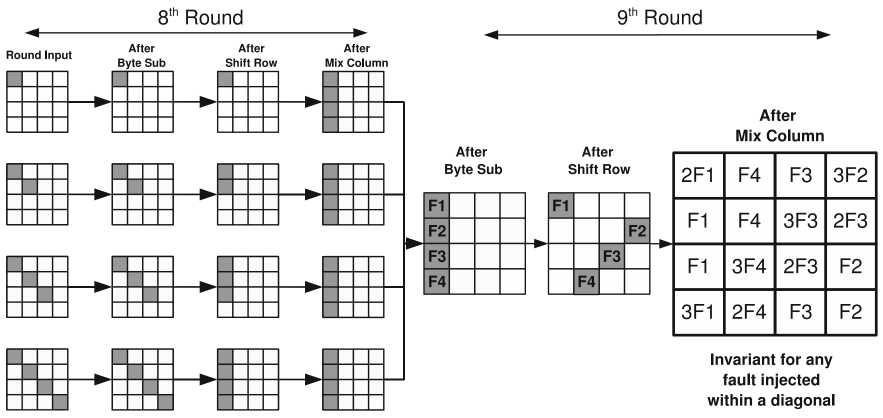
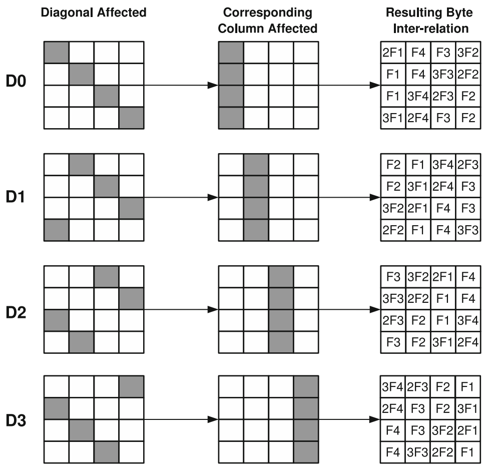


Fig. 10 Equivalence of different kinds of faults induced in diagonal D_0 at the input of eighth round of AES

Fig. 11 Byte inter-relations at the end of ninth round corresponding to different diagonals being faulty



As before these equations reduce the AES key to an average size of 2^{32}. If the attacker is unaware of the exact diagonal, he can repeat for all the above four sets of equations, and the key size will still be $2^{32} \times 4 = 2^{34}$, which can be brute forced feasibly with present day computation power.

Next, we consider briefly the cases when the faults spread to more than one diagonal.

7.2 DFA According to Fault Model *DM*1

In Fig. 12, we observe the propagation of faults when the diagonals, D_0 and D_1 are affected at the input of the ninth round MixColumns.

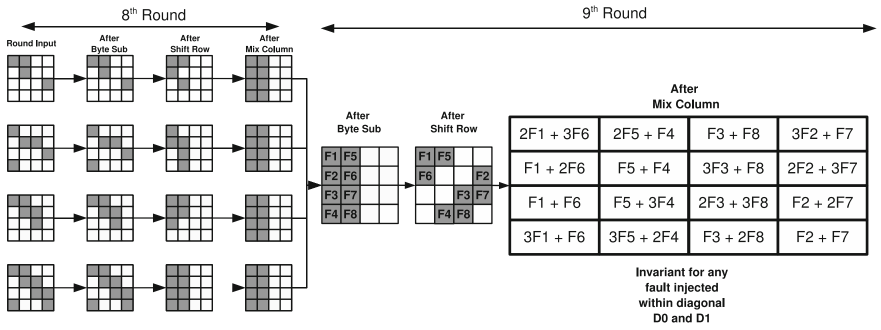


Fig. 12 Fault propagation if diagonals D_0 are D_1 are affected

We observe that the nature of the faults in the state matrix at the input of the ninth round MixColumns and hence at the output remains invariant for all possible faults in these two diagonals. This property is exploited to develop equations which are used to retrieve the correct key.

We denote the fault values in the first column of the output of the ninth round MixColumns by a_0, a_1, a_2, a_3, where each a_i is a byte $0 \leq i \leq 3$. Then using the inter-relationships among the faulty bytes one can easily show that:

$$a_1 + a_3 = a_0$$
$$2a_1 + 3a_3 = 7a_2$$

We can express a_0, a_1, a_2, a_3 in terms of the fault free ciphertext (CT), faulty ciphertext (CT^*) and 4 bytes of the tenth round key ($\mathbf{K_{10}}$). The equations reduce the key space of 4 bytes of the key to 2^{16}. Similarly, performing the analysis for other columns, helps to reduce the AES key to a size of $(2^{16})^4 = 2^{64}$. Using two such faulty inductions it is expected that the unique key is returned.

Depending on the combination of two diagonals affected out of the four diagonals, there are six such sets of equations. Hence even in such case, the attacker reduces the AES key space to 6 possible keys, which he can easily brute force.

In the next section, we present an attack strategy if the fault gets spread to atmost three diagonals. This fault model, $DM2$ thus covers the first two models of fault.

7.3 DFA According to Fault Model DM2

In Fig. 13, we observe the propagation of faults when the diagonals, D_0, D_1 and D_2 are affected.

From Fig. 13, we note that for all possible faults corrupting the diagonals D_0, D_1 and D_2, the nature of the faults at the input of the ninth round MixColumns is an invariant. The fault nature at the output of the ninth round MixColumns is as seen

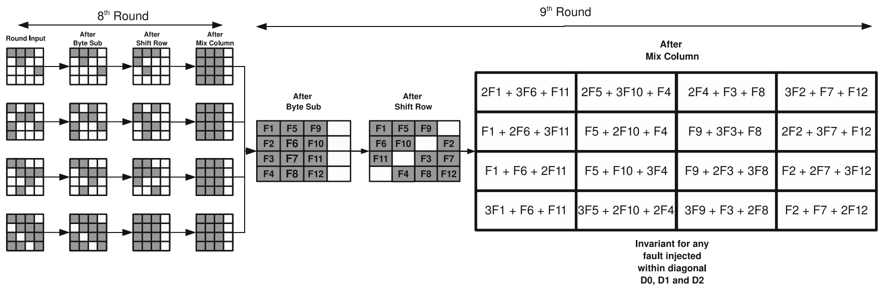


Fig. 13 Fault propagation if diagonals D_0, D_1 and D_2 are affected

in the figure, also an invariant. We denote the fault values in the first column of the output of the ninth round MixColumns by a_0, a_1, a_2, a_3, where each a_i is a byte $0 \leq i \leq 3$.

The following equation can be obtained by observing the inter-relationships (refer Fig. 12):

$$11a_0 + 13a_1 = 9a_2 + 14a_3$$

As before in case of faults modeled by $DM1$, we can express a_0, a_1, a_2, a_3 in terms of the fault free ciphertext (CT), faulty ciphertext (CT') and the tenth round key ($\mathbf{K_{10}}$). One equation reduces 4 bytes of the key to 2^{24} values. We can have similar equations for each of the remaining three columns of the state matrix after the ninth round MixColumns, and thus the AES key space reduces to an expected value of $(2^{24})^4 = 2^{96}$. However, using four faults and taking the intersection of the key space, it is expected that the key space reduces to a unique value.

It may be noted that when the faults occur according to the model $DM3$, that is all the four diagonals are affected, the DFA fails.

8 Extension of the DFA to Other Variants of AES

In the previous sections we described DFAs using different fault models on AES with 128-bit key. However, AES has two more variants: AES-192 and AES-256 with key length 192 and 256 bits. These to variants of AES follows different key scheduling. If we observe the key scheduling algorithm, we see that for AES-192 and AES-256, last round is not sufficient to retrieve the master key. It requires to retrieve the last two round keys rather any two consecutive round keys. For the sake of simplicity we consider the last two round keys. In case of AES-192, the last round key and the last two columns of penultimate round key is sufficient. Because the first two columns of penultimate round key can be directly derived from the final round key.

The first complete DFA on AES-192 and AES-256 was proposed in [33]. The proposed attacks were based on two different fault models which requires 6 and 3000 pairs of fault free and faulty ciphertexts. A new attack was proposed in [52] which first time exploited the relations between the round keys of the key scheduling algorithm. The attack on AES-192 required three pairs of correct and faulty ciphertexts and the attack on AES-256 required two pairs of correct and faulty ciphertexts and two pairs of correct and faulty plaintexts. The attack was further improved in [29] where the DFA on AES-192 required two pairs of fault free and faulty ciphertexts, and on AES-256 required three pairs of fault free and faulty ciphertexts. Recently a DFA on AES-256 was proposed in [4], which required two pairs of fault-free and faulty ciphertexts and a brute-force search of 16 bits with attack time complexity of 2^{32}. This is the best known attack on AES-256 till date. More details of the attacks on the other versions of AES can be obtained from [14].

9 DFA of AES Targeting the Key-Schedule

In the previous sections we described how an induced difference at the state of a particular round of AES can be exploited to reveal the secret key. In order to protect AES from such attacks a designer has to use some countermeasures which will not allow the attacker to induce faults in AES round operations. Even if fault is induced, the attacker will not be able to get the faulty ciphertexts to apply a DFA. Subsequently, the attackers have developed new attacking technique known as DFA on AES key schedule which work even if the rounds of the AES are protected against faults. In this kind of DFAs, faults are induced at the round keys. Therefore, even if the rounds are protected against DFA, the attack will work as the protection will not be able to distinguish between a fault-free round key and a faulty round key.

However, the DFA on AES key schedule are more challenging than DFA on AES state. A difference induced in a round key will spread to more number of bytes in the subsequent round keys during the key schedule operation, which in turn creates more number of unknown variables in the differential equations. Therefore, the differential equations are more complex than the differential equations in a DFA on AES state.

The first complete DFA on AES key schedule was proposed in [13]. The attack was targeted on AES-128 and required less than 30 pairs of fault-free and faulty ciphertexts. An improved attack in [53] showed that a DFA on AES key schedule is possible using two pairs of fault-free and faulty ciphertexts and a brute-force search of 48-bit. Subsequently, there are two more attacks proposed in [32] and [31] using two pairs of fault-free and faulty ciphertexts each. Most optimum attack on AES key schedule was proposed in [3, 5], which required only one pair of fault-free and faulty ciphertexts. The attack used a complex divide and conquer strategy to solve the differential equations. The most recent attacks on AES, both state and key schedule, can be found in [7].

Table 2 Comparison of CED techniques

CED		†Fault coverage	CED/original	
			Throughput (%)	Hardware (%)
Hardware redundancy	[18]	91 %	∼100	∼200
	[42]	25 %	∼100	126
Time redundancy	[35]	99.9 %	∼100	136
	[46]	100 %	50–90.9	102.3
Information redundancy	[55]	48–53 %	∼100	122.3
	[9]	99.997 %	67.86	188.9
	[41]	99.20 %	–	137.35
	[19]	$1 - 2^{-56}$	87	177
Hybrid redundancy	[22]	100 %	73.45	197.6
	[48]	–	85.6	188.9

† Fault coverage are based on multiple bit random fault model, ∼ Estimated values: authors did not give precise figures

10 CED for AES

Faults that occur in VLSI chips are classified into two categories: transient faults that eventually die away and permanent faults. The origin of these faults could be internal phenomena in the system, such as threshold changes, shorts, opens, etc., or external influences, such as electromagnetic radiation. These faults affect the memory as well as the combinational parts of a circuit and are detected using concurrent error detection (CED) [50]. Cryptographic chips are sensitive to faults in the hardware. A small number of excited faults can cause a large number of output bits of AES to be faulty [9]. As previously explained, attackers have injected faults into cryptographic circuits to steal secret information. Previous work on CED can be classified into four types of redundancy: hardware, time, information, and hybrid redundancy. Table 2 shows a comparison of different CED techniques based on fault coverage, throughput and hardware utilization. The fault coverage is obtained from the multiple bit random fault model. The throughput and hardware are the ratio between the CED implementation and the original.

10.1 Hardware Redundancy

Hardware redundancy duplicates the function and detects faults by comparing the outputs of two copies.

In [18], the authors propose a novel hardware redundancy technique for AES to detect faults. Because an attacker can potentially inject the same faults to both of the AES circuits, the straightforward hardware redundancy can be bypassed by the attacker. As shown in Fig. 14, the idea is to mix byte states between the operations in

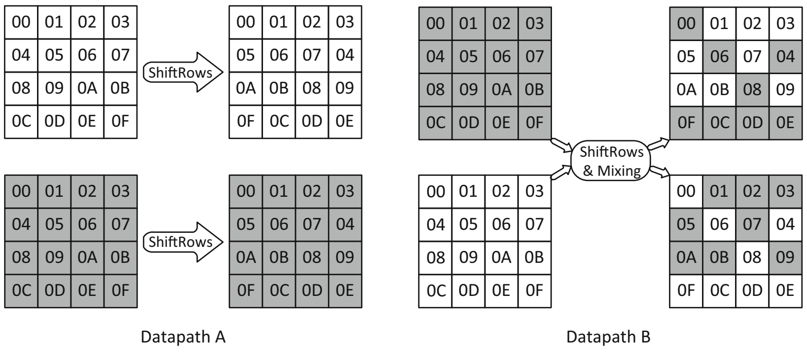


Fig. 14 Datapath mixing (byte level)

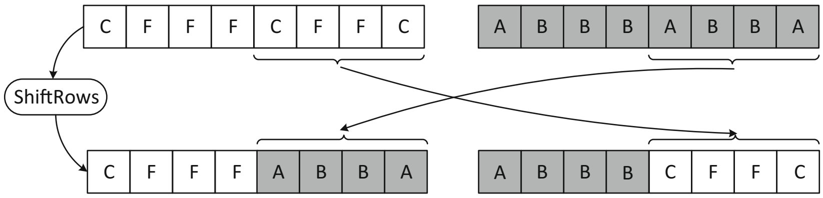


Fig. 15 Datapath mixing (bit level)

two pieces of hardware, in different ways and at different locations. The mixing can be done at byte level or bit level as shown in Figs. 14 and 15. Because the attacker does not know how the circuits are mixed, the attacker needs to reverse engineer the circuit layout to figure out where to inject the faults. Although this technique adds an extra layer of obscurity on top of the straightforward hardware redundancy, its effectiveness is yet to be proven. It also requires significant changes to the AES datapath. Because the entire hardware is duplicated, hardware redundancy has high fault coverage, low fault detection latency, and low performance overhead. However, because one extra piece of hardware and comparison circuitries are needed, the hardware overhead is approximately 200 % as shown in Table 2. It provides 91 % fault coverage and the performance is close to the original implementation.

To reduce the hardware overhead, [42] proposes a partial hardware redundancy technique as shown in Fig. 16. This technique focuses on parallel AES architecture and S-box protection. The idea is to add an additional S-box to every set of four S-boxes, and perform two tests of every S-box per encryption cycle (10 rounds). Although the hardware overhead is reduced to 26 %, this process has a fault coverage of 25 % at a certain clock cycle, because it can only check one S-box among every four in one clock cycle.

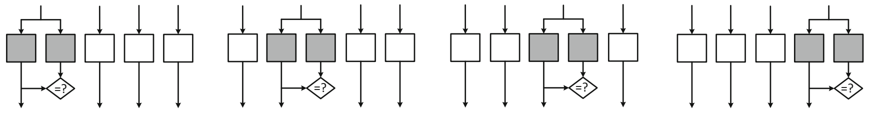


Fig. 16 Partial hardware redundancy

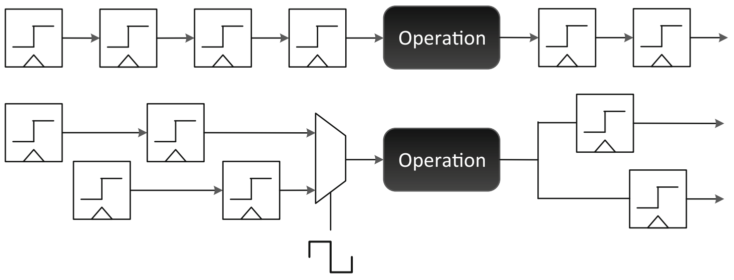


Fig. 17 Candidate operation for DDR application, and operation computing after DDR application

10.2 Time Redundancy

Time redundancy computes the same input twice using the same function and compares the results.

In [35], the authors propose time redundancy with a Double-Data-Rate (DDR) mechanism as shown in Figs. 17 and 18. The pipelined AES data path logic is partitioned into two classes, where nonadjacent stages in the pipeline are driven by two opposite clock signals. The DDR architecture allows us to halve the number of clock cycles per round, though maybe with a light impact on clock frequency as compared to a design without protection. This takes advantage of the free cycles for recomputing the round on the same hardware. Two successive round operation outputs obtained from two copies of the same input data are checked for possible mismatches. It shows an almost maximal fault coverage on the datapath at the cost of 36 % hardware overhead. Under some conditions, this technique allows the encryption to be computed twice without affecting the global throughput. However, this technique becomes difficult to implement as technology scales.

A technique that is suited for any pipeline-based block cipher design is proposed in [46]. The key idea is to use different pipeline stages to check against each other by shifting the computation from one stage to another. Let us assume the pipeline has n stages as shown in Fig.19. In the normal computation, the plaintext will be computed by the first stage and the then the second stage and so on. The nth stage will produce the ciphertext. In the CED computation, the plaintext will be computed by the nth stage, and then the output of the nth stage will be computed by the first stage. Therefore, the output of the $(n-1)$th stage is the ciphertext and it will be compared with the previous ciphertext. Compared to the original design, this CED provides a throughput of 50–90.0 %, depending on the frequency of the redundant check from every one to ten rounds. Hardware overhead is only 2.3 %.

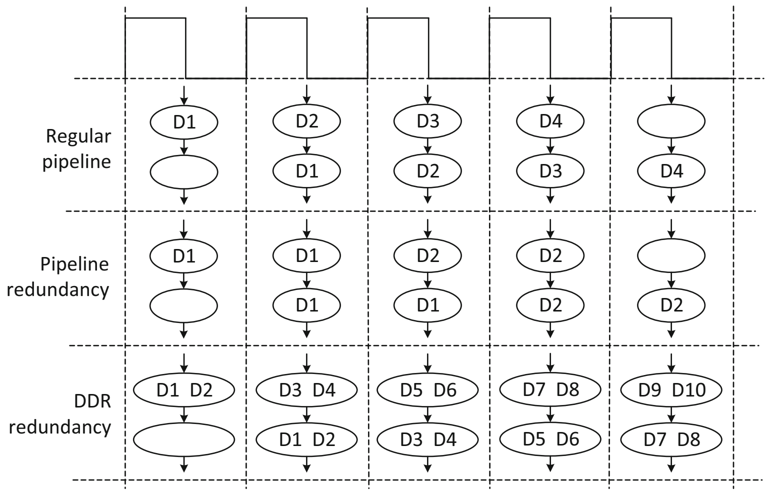
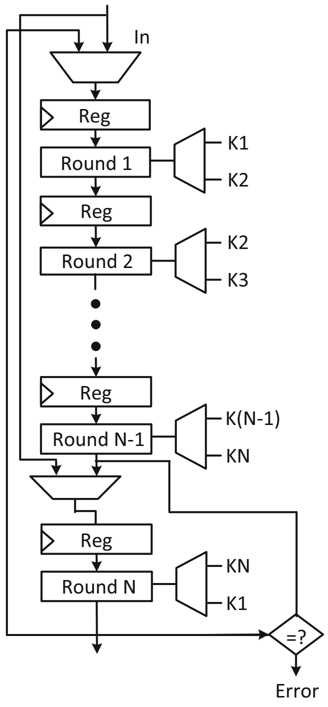


Fig. 18 Scheduling of a regular pipeline, pipeline redundancy and the DDR redundancy (D_x is data token X)

Fig. 19 Sliced architecture



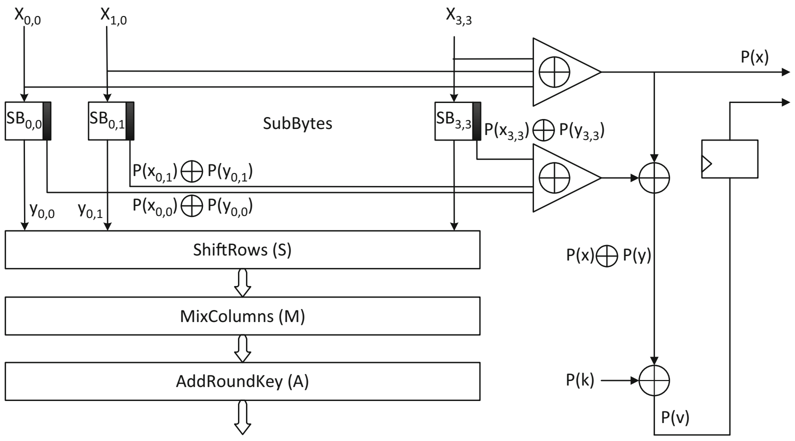


Fig. 20 Parity-1 CED

10.3 Information Redundancy

Information redundancy techniques are based on error detecting codes (EDC). A few check bits are generated from the input message; then they propagate along with the input message and are finally validated when the output message is generated. Parity code and robust code are proposed for CED in various research [9, 24, 25, 27, 37–39].

10.3.1 Parity-1

A technique in which a parity bit is used for the entire 128-bit state matrix is developed in [55]. The parity bit is checked once for the entire round as shown in Fig. 20. This approach targets low-cost CED. Parity-1 is based on a general CED design for Substitution Permutation Networks (SPN) [23], in which the input parity of SPN is modified according to its processing steps into the output parity and compared with the output parity of every round. The authors adapt this general approach to develop a low-cost CED. First, they determine the parity of the 128-bit input using a tree of XOR gates. Then for the nonlinear S-box, inversion in $GF(2^8)$ and a linear affine transformation. They add one additional binary output to each of the 16 S-boxes. This additional S-box output computes the parity of every 8-bit input and the parity of the corresponding 8-bit output.

Each of the modified S-boxes is 8-bit by 9-bit. The additional single-bit outputs of the 16 S-boxes are used to modify the input parity for SubBytes. Because ShiftRows implements a permutation, it does not change the parity of the entire state matrix from its input to output. MixColumns does not change the parity of the state matrix from inputs to outputs either. Moreover, MixColumns does not

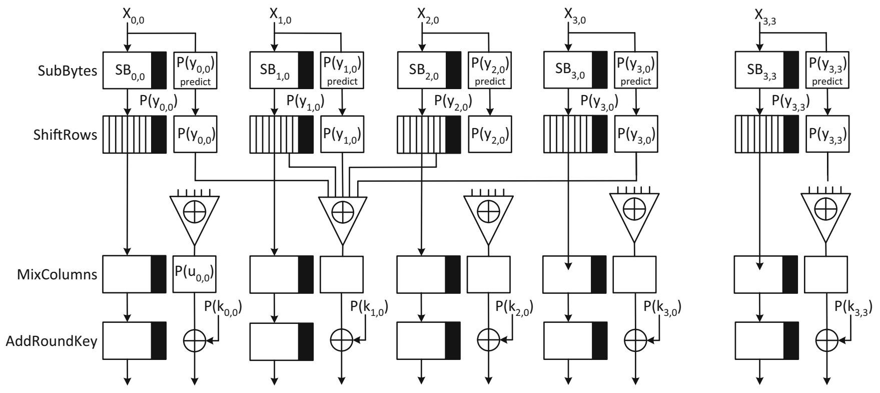


Fig. 21 Parity-16 CED

change the parity of each column. Finally, the bit-wise XOR of the 128-bit round key needs a parity modification by a single precomputed parity bit of the round key. Because the output of a round is the input to the next round, the output parity of a round can be computed with the same hardware for computing the input parity of the previous round.

Although this technique has only 22.3 % hardware overhead, it has 48–53 % fault coverage for multiple bit random fault model.

10.3.2 Parity-16

Parity-16 is first proposed in [9]. In this technique, each predicted parity bit is generated from an input byte. Then, the predicted parity bits and actual parity bits of output are compared to detect the faults.

In [9], the authors propose the use of a parity bit that is associated with each byte of the state matrix of a 128-bit iterated hardware implementation with LUT-based S-Boxes as shown in Fig. 21. Predicted parity bits on S-Box outputs are stored as additional bits in the ROMs (nine bits instead of eight in the original S-Boxes). In order to detect errors in the memory, the authors propose increasing each S-box to 9-bit by 9-bit in such a way that all the ROM words addressed with a wrong input address (i.e. S-Boxes input with a wrong associated parity), deliberately store values with a wrong output parity so that the CED will detect the fault. As before, the parity bit associated with each byte is not affected by ShiftRows. In Parity-1, the global parity bit on the 128 bits remains unchanged after MixColumns. Conversely, at the byte level, the parity after MixColumns is affected. Therefore, parity-16 requires the implementation of prediction functions in MixColumns. Finally, the parity bits after AddRoundKey are computed as before, by adding the current parity bits to those of the corresponding round key. This technique incurs 88.9 % hardware overhead because of the LUT size is doubled. The throughput is 67.86 % of the original.

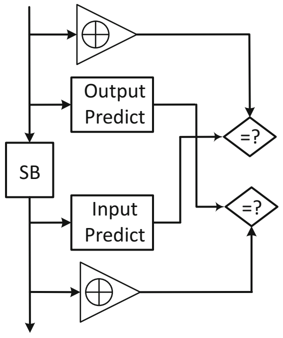


Fig. 22 Parity-32 CED

10.3.3 Parity-32

As shown in Fig. 22, a technique that strengthen fault detection on S-Boxes is proposed in [41]. With respect to parity-16, this technique still uses one parity bit for each byte in all the operations except SubBytes. It adds one extra parity bit for each S-box in SubBytes; one parity bit for the input byte and one for the output byte. The actual output parity is compared with the predicted output parity, and the actual input parity bit is compared with the predicted input parity. It has 37.35 % hardware overhead and 99.20 % fault coverage.

10.3.4 Parity Code vs Residue Code

In [12], the authors try to apply EDCs in a systematic way by investigating 11 different symmetric key encryption algorithms. They study the feasibility of two EDCs for different cryptographic operations mainly from a hardware overhead point of view. For EDCs, parity and residue code are considered. Cryptographic algorithms are divided into XOR, AND/OR, finite field addition/subtraction (mod n), finite field multiplication [mod n and mod $G(x)$], expansion, S-Box, word rotation, word shift, and permutation. The conclusion is that parity code is superior to residue code in logic operations, while residue code is more feasible for arithmetic operations. The authors recommend one parity bit per byte for the AES error detection from a low-cost and high fault coverage point of view.

10.3.5 Robust Code

Robust codes is first proposed in [19]. The idea is to use non-linear EDC instead of linear. Robust codes can be used to extend the error coverage of any linear prediction technique for AES. The advantage of non-linear EDC is because it has uniform fault coverage. If all the data vectors and error patterns are equiprobable, then the probability of injecting an undetectable fault is the same for all of them.

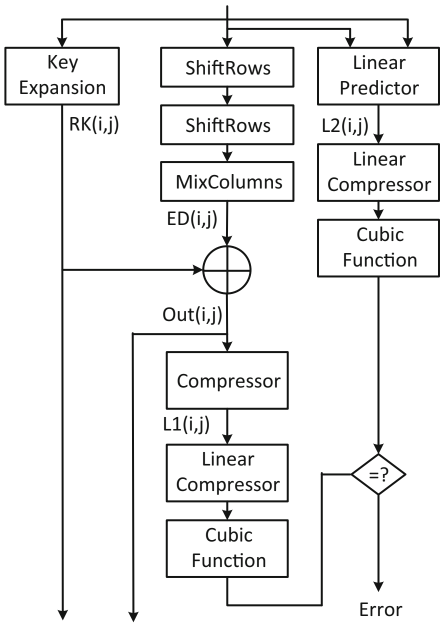


Fig. 23 Robust code

The architecture of AES with robust protection is presented in Fig. 23. In this architecture, two extra unit are needed. One is the prediction unit at the round input, and it includes a linear predictor, a linear compressor, and a cubic function. The other one is the comparison unit at the output of the round, and it includes a compressor, a linear compressor, and a cubic function This architecture protects the encryption and decryption as well as key expansion.

Let us first introduce the prediction unit. A linear predictor and linear compressor is designed to generate an 32-bit output, and we call them the linear portion in the rest of the chapter. The output of the linear portion is linearly related to the output of the round of AES as shown in Fig. 23. They offer a relatively compact design compared to the original round of AES. They simplify the round function by XORing the bytes in the same column. The effect of MixColumns is removed by the linear portion. As a result, the linear portion is greatly simplified as it no longer needs to perform multiplication associated with the MixColumns or InvMixColumns. For the cubic function, the input of is cubed in $GF(2^r)$ to produce the r-bit output, and thus it is non-linear with respect to the output of the round.

In the comparison unit, the compressor and the linear compressor are designed to generate a 32-bit output from the 128-bit round output. The bytes in the same column of the output is XORed. Again, the 32-bit output is cubed in the cubic function to generate r-bit output. This output is then compared with the output from the prediction unit.

This technique provides $1 - 2^{-56}$ fault coverage, and it has a 77 % hardware overhead.

Although these countermeasures can thwart DFA, the designer needs to be cautious when implementing information redundancy-based techniques. Because they increase the correlation of the circuit power consumption with the processed data, the side channel leakage is also increased [34].

10.4 Hybrid Redundancy

In [20–22], the authors consider CED at the operation, round, and algorithm levels for AES. In these schemes, an operation, a round, or the encryption and decryption are followed by their inverses. To detect faults, the results are compared with the original input.

The underlying assumption is that a complete encryption device operating in ECB mode consists of encryption and decryption modules, Thus, a low-cost and low-latency systematic CED is proposed for encryption and decryption datapaths. They describe algorithm-, round-, and operation-levels CEDs that exploit the inverse relationship properties of AES. Because AES uses the same set of round keys for both encryption and decryption, they can be generated a priori, stored in the key RAM, and retrieved in any order depending upon whether encryption or decryption is in progress. They then extend the proposed techniques to full duplex mode by trading off throughput and CED capability.

As shown in Fig. 24a, the algorithm-level CED approach exploits the inverse relationship between the entire encryption and decryption. Plaintext is first processed

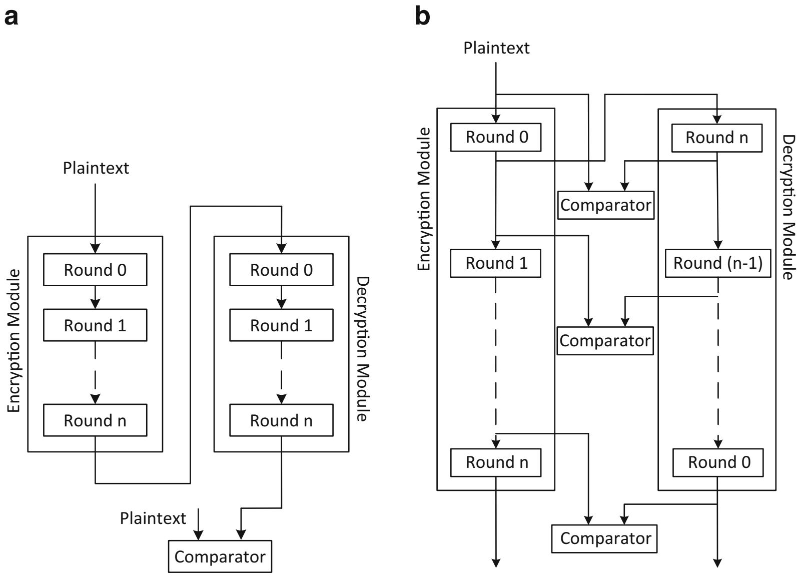


Fig. 24 Hybrid redundancy. (**a**) Algorithm level. (**b**) Round level

by the encryption module. After the ciphertext is available, the decryption module is enabled to decrypt the ciphertext. While the decryption module is decrypting ciphertext, the encryption module can process the next block of data or be idle. A copy of plaintext is also temporarily stored in a register. The output of decryption is compared with this copy of the input plaintext. If there is a mismatch, an error signal will be raised, and the faulty ciphertext will be suppressed.

For AES, the inverse relationship between encryption and decryption exists at the round level as well. Any input data passed successively through one encryption round are recovered by the corresponding decryption round.

For almost all the symmetric block cipher algorithms, the first round of encryption corresponds to the last round of decryption; the second round of encryption corresponds to the next-to-the-last round of decryption, and so on. Based on this observation, CED computations can also be performed at the round level. At the beginning of each encryption round, the input data is stored in a register before being fed to the round module. After one round of encryption is finished, output is fed to the corresponding round of decryption. Then, the output of the decryption round is compared with the input data saved previously. If they are not the same, encryption is halted and an error signal is raised. Encryption with round-level CED is shown in Fig. 24b.

Depending on the block ciphers and their hardware implementation, each round may consume multiple clock cycles. Each round can be partitioned into operations and subpipelined to improve performance. Each operation can consume one or more clock cycles, such that the operations of encryption and corresponding operations of decryption satisfy the inverse relationship. As shown in Fig. 25a, applying input data to the encryption operation and the output data of the encryption operation to the corresponding inverse operation in decryption yields the original input data. The boundary on the left shows the rth encryption round while the boundary on the right shows the $(n - r + 1)$th decryption round, where r is the total number of rounds in encryption/decryption. Figure 25a also shows that the first operation of the encryption round corresponds to the mth operation of the decryption round, which is the last operation of the decryption round. Output from operation one of encryption is fed into the corresponding inverse operation m of decryption.

Although these techniques has close to 100 % fault coverage, their throughput is 73.45 % of the original AES in half-duplex mode. It can suffer from more than 100 % throughput overhead if the design is in full-duplex mode. The hardware overhead is minimal if both encryption and decryption are on the chip. However, if only encryption or decryption is used in the chip, it will incur close to 100 % hardware overhead.

To reduce the hardware overhead of the previous technique, [48] proposes a novel hardware optimization, and thus, reduced the hardware utilization significantly. Figure 25b shows the architecture. It divides a round function block into two sub-blocks and uses them alternatively for encryption (or decryption) and error detection. Therefore, no extra calculation block is needed, even though only a pipeline register, a selector and a comparator are added. The number of operating cycles is doubled, but the operating frequency is boosted because the round function

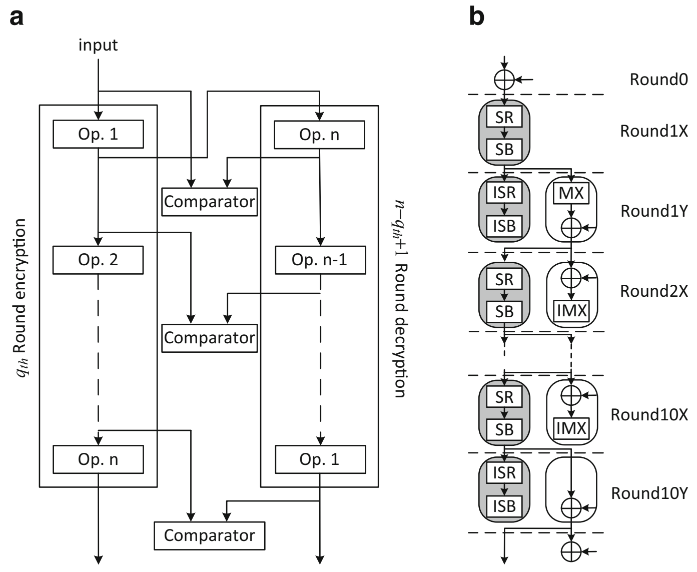


Fig. 25 Hybrid redundancy. (**a**) Operation level. (**b**) Optimized version

block in the critical path is halved. Therefore, the technique provides 85.6 % throughput compared to 73.45 % in the previous one. The hardware overhead also decrease from 97.6 to 88.9 %.

10.5 Other Techniques

In [24, 37, 39], parity is obtained for S-box implementation in finite field arithmetic with polynomial basis. In [25, 27, 38], parity is obtained for S-box implementation in finite field arithmetic with normal basis. In [26], an AES parity detection method with mixed basis is proposed. All these parity schemes share the same limitation. If an even number of faults occur in the same byte, none of these schemes can detect them.

While traditional CED techniques have their strengths and limitations, techniques based on algorithmic invariances can offer new tradeoff choices. An invariance-based CED is proposed in [17]. It utilizes a round-level invariance of the AES and checks for the invariance property. Because the invariance does not constrain the input pattern of the round, it is very flexible and the fault coverage is very high. Because the invariance property is a permutation property, the hardware overhead only includes the comparator and muxes that are used to select the regular or the permuted datapath.

11 Conclusion

The chapter first shows that faults can be threatening to the security of modern day block ciphers. Taking the present day standard block cipher, AES, as an example the chapter shows techniques for performing differential fault analysis of block ciphers. The attacks have been described on the assumptions of various types of fault models: namely bit, single byte, and multiple byte. The fault analysis of AES using only one instance of a random single byte fault have been shown to reduce the key space of AES to on an average 28 values through an attack with time complexity 2^{30}. Likewise, the chapter also shows even when the faults spread to multiple bytes, corrupting 12 out of 16 bytes of the AES state matrix, DFA of AES is feasible. All these attacks unfurl that DFA of block ciphers is a very strong attack model, where even random faults can completely collapse the security of even very strong block cipher algorithms.

The practicality of such fault models and the requirement of such small number of faulty computations, emphasizes the need for CED techniques. We present a study on CED techniques deployed for detecting DFA of AES hardware implementation. We classify the CED techniques into hardware, time, information, and hybrid redundancies. We also summarize the overheads and fault coverage of the countermeasures. Hardware and hybrid redundancies provides high security and reliability but the hardware overhead is also high. Time redundancy is low cost but also decrease the performance. It also has limitations when there are permanent faults and long transient faults. Information redundancy provides high reliability with relatively low hardware and performance overhead. Although these countermeasures can thwart DFA, the designer needs to be cautious in implementation. Some techniques may enable backdoors for other types of attacks. For example, information redundancy-based techniques tend to increase the correlation of the circuit power consumption, with the processed data increasing the side channel leakage. While traditional CED techniques have their strengths and limitations, techniques based on algorithmic invariances can offer new tradeoff choices.

Acknowledgements The fourth author would like to acknowledge Indo-US S&T Forum for providing fellowship to support the above collaboration. The chapter was partially written during his visit as a researcher under the Indo-USSTF Fellowship to NYU-Poly, USA in 2012.

References

1. Agoyan, M., Dutertre, J.M., Mirbaha, A.P., Naccache, D., Ribotta, A.L., Tria, A.: How to flip a bit? In: IEEE International On-Line Testing Symposium (IOLTS), pp. 235–239 (2010)
2. Agoyan, M., Dutertre, J.M., Naccache, D., Robisson, B., Tria, A.: When clocks fail: on critical paths and clock faults. In: International Conference on Smart Card Research and Advanced Application (CARDIS), pp. 182–193 (2010)
3. Ali, S.S., Mukhopadhyay, D.: A differential fault analysis on AES key schedule using single fault. In: Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 35–42 (2011)

4. Ali, S.S., Mukhopadhyay, D.: An improved differential fault analysis on AES-256. In: International Conference on Cryptology in Africa (AFRICACRYPT), pp. 332–347 (2011)
5. Ali, S.S., Mukhopadhyay, D.: Differential fault analysis of AES-128 key schedule using a single multi-byte fault. In: International Conference on Smart Card Research and Advanced Applications (CARDIS), pp. 50–64 (2011)
6. Ali, S.S., Mukhopadhyay, D., Tunstall, M.: Differential fault analysis of AES using a single multiple-byte fault. Cryptology ePrint Archive, Report 2010/636 (2010). http://eprint.iacr.org/
7. Ali, S.S., Mukhopadhyay, D., Tunstall, M.: Differential fault analysis of AES: towards reaching its limits. J. Cryptogr. Eng. **3**(2), 73–97 (2013). doi:10.1007/s13389-012-0046-y. http://dx.doi.org/10.1007/s13389-012-0046-y
8. Barenghi, A., Hocquet, C., Bol, D., Standaert, F.X., Regazzoni, F., Koren, I.: Exploring the feasibility of low cost fault injection attacks on sub-threshold devices through an example of a 65nm AES implementation. In: Proceedings of Workshop RFID Security Privacy, pp. 48–60 (2011)
9. Bertoni, G., Breveglieri, L., Koren, I., Maistri, P., Piuri, V.: Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. IEEE Trans. Comput. **52**(4), 492–505 (2003)
10. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Proceedings of Eurocrypt. Lecture Notes in Computer Science, vol. 1233, pp. 37–51 (1997)
11. Blömer, J., Seifert, J.P.: Fault based cryptanalysis of the Advanced Encryption Standard (AES). In: Financial Cryptography, pp. 162–181 (2003)
12. Breveglieri, L., Koren, I., Maistri, P.: An operation-centered approach to fault detection in symmetric cryptography ciphers. IEEE Trans. Comput. **56**, 635–649 (2007)
13. Chen, C.N., Yen, S.M.: Differential fault analysis on AES key schedule and some coutnermeasures. In: Australasian Conference on Information Security and Privacy (ACISP), pp. 118–129 (2003)
14. Debdeep, M., Rajat Subhra, C.: Hardware Security: Designs, Threats, and Safeguards, CRC Press (2014)
15. Dusart, P., Letourneux, G., Vivolo, O.: Differential fault analysis on AES. In: Cryptology ePrint Archive, pp. 293–306 (2003)
16. Giraud, C.: DFA on AES. In: IACR e-print archive 2003/008, p. 008 (2003). http://eprint.iacr.org/2003/008
17. Guo, X., Karri, R.: Invariance-based concurrent error detection for advanced encryption standard. In: ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 573–578 (2012)
18. Joye, M., Manet, P., Rigaud, J.: Strengthening hardware AES implementations against fault attack. IET Inf. Secur. **1**, 106–110 (2007)
19. Karpovsky, M., Kulikowski, K.J., Taubin, A.: Differential fault analysis attack resistant architectures for the advanced encryption standard. In: World Computer Congress on Smart Card Research and Advanced Applications VI (CARDIS), pp. 177–192 (2004)
20. Karri, R., Wu, K., Mishra, P., Kim, Y.: Concurrent error detection of fault-based side-channel cryptanalysis of 128-bit symmetric block ciphers. In: Design Automation Conference (DAC), pp. 579–585 (2001)
21. Karri, R., Wu, K., Mishra, P., Kim, Y.: Fault-based side-channel cryptanalysis tolerant rijndael symmetric block cipher architecture. In: IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT) pp. 427–435 (2001)
22. Karri, R., Wu, K., Mishra, P., Kim, Y.: Concurrent error detection schemes of fault based side-channel cryptanalysis of symmetric block ciphers. IEEE Trans. Comput. Aided Des. **21**(12), 1509–1517 (2002)
23. Karri, R., Kuznetsov, G., Goessel, M.: Parity-based concurrent error detection of substitution-permutation network block ciphers. In: International Workshop on Cryptographic Hardware and Embedded Systems (CHES), pp. 113–124 (2003)
24. Kermani, M.M., Reyhani-Masoleh, A.: Parity prediction of S-box for AES. In: Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 2357–2360 (2006)

25. Kermani, M.M., Reyhani-Masoleh, A.: A Low-cost S-box for the advanced encryption standard using normal basis. In: IEEE International Conference on Electro/Information Technology (EIT), pp. 52–55 (2009)
26. Kermani, M.M., Reyhani-Masoleh, A.: A high-performance fault diagnosis approach for the AES subbytes utilizing mixed bases. In: Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 80–87 (2011)
27. Kermani, M.M., Reyhani-Masoleh, A.: A low-power high-performance concurrent fault detection approach for the composite field S-Box and inverse S-Box. IEEE Trans. Comput. **60**(9), 1327–1340 (2011)
28. Khelil, F., Hamdi, M., Guilley, S., Danger, J.L., Selmane, N.: Fault analysis attack on an AES FPGA implementation. In: ESRGroups, pp. 1–5 (2008)
29. Kim, C.H.: Differential fault analysis against AES-192 and AES-256 with minimal faults. In: Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 3–9 (2010)
30. Kim, C.H.: Differential fault analysis of AES: toward reducing number of faults. Cryptology ePrint Archive, Report 2011/178 (2011). http://eprint.iacr.org/
31. Kim, C.H.: Improved differential fault analysis on AES key schedule. IEEE Trans. Inf. Forensics Secur. **7**(1), 41–50 (2012)
32. Kim, C.H., Quisquater, J.J.: New differential fault analysis on AES key schedule: two faults are enough. In: International Conference on Smart Card Research and Advanced Applications (CARDIS), pp. 48–60 (2008)
33. Li, W., Gu, D., Wang, Y., Li, J., Liu, Z.: An extension of differential fault analysis on AES. In: International Conference on Network and System Security (NSS), pp. 443–446 (2009)
34. Maingot, V., Leveugle, R.: Influence of error detecting or correcting codes on the sensitivity to DPA of an AES S-Box. In: International Conference on Signals, Circuits and Systems (ICSES), pp. 1–5 (2009)
35. Maistri, P., Leveugle, R.: Double-data-rate computation as a countermeasure against fault analysis. IEEE Trans. Comput. **57**(11), 1528–1539 (2008)
36. Moradi, A., Shalmani, M.T.M., Salmasizadeh, M.: A generalized method of differential fault attack against AES cryptosystem. In: International Workshop on Cryptographic Hardware and Embedded Systems (CHES), pp. 91–100 (2006)
37. Mozaffari-Kermani, M., Reyhani-Masoleh, A.: Parity-based fault detection architecture of s-box for advanced encryption standard. In: IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT), pp. 572–580 (2006)
38. Mozaffari-Kermani, M., Reyhani-Masoleh, A.: A lightweight concurrent error detection scheme for the AES S-boxes using normal basis. In Proceedings of Cryptographic Hardware and Embedded Systems (CHES), pp. 113–129 (2008)
39. Mozaffari-Kermani, M., Reyhani-Masoleh, A.: A lightweight high-performance fault detection scheme for the advanced encryption standard using composite field. IEEE Trans. VLSI Syst. **19**(1), 85–91 (2011)
40. Mukhopadhyay, D.: An improved fault based attack of the advanced encryption standard. In: Cryptology in Africa, AFRICACRYPT, pp. 421–434 (2009)
41. Natale, G.D., Flottes, M.L., Rouzeyre, B.: A novel parity bit scheme for SBox in AES circuits. In: IEEE Design and Diagnostics of Electronic Circuits and Systems (DDECS '07), pp. 1–5 (2007)
42. Natale, G.D., Flottes, M.L., Rouzeyre, B.: On-line self-test of AES hardware implementation. In: Workshop on DSN (2007)
43. National Institute of Stardards and Technology (NIST).: Advanced Encryption Standard (AES). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf (2001)
44. Nyberg, K.: Differentially uniform mappings for cryptography. In: Advances in Cryptology - EUROCRYPT'93, pp. 55–64 (1993)
45. Piret, G., Quisquater, J.: A differential fault attack technique against SPN structures, with application to the AES and khazad. In: Proceedings of Cryptographic Hardware and Embedded Systems (CHES), pp. 77–88 (2003)

46. Rajendran, J., Borad, H., Mantravadi, S., Karri, R.: Sliced: slide-based concurrent error detection technique for symmetric block cipher. In: International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 70–75 (2010)
47. Saha, D., Mukhopadhyay, D., RoyChowdhury, D.: A diagonal fault attack on the advanced encryption standard. Cryptology ePrint Archive, Report 2009/581 (2009). http://eprint.iacr.org/
48. Satoh, A., Sugawara, T., Homma, N., Aoki, T.: High-performance concurrent error detection scheme for AES hardware. In: Cryptographic Hardware and Embedded Systems (CHES), pp. 100–112 (2008)
49. Selmane, N., Guilley, S., Danger, J.L.: Practical setup time violation attacks on AES. In: European Dependable Computing Conference, pp. 91–96 (2008)
50. Siewiorek, D.P., Swarz, R.S.: Reliable Computer Systems: Design and Evaluation, 3rd edn. A K Peters/CRC Press, A. K. Peters, Ltd. (1998)
51. Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In: Proceedings of Cryptographic Hardware and Embedded Systems (CHES), pp. 2–12 (2002)
52. Takahashi, J., Fukunaga, T.: Differential fault analysis on AES with 192 and 256-bit keys. Cryptology ePrint Archive, Report 2010/023 (2010). http://eprint.iacr.org/
53. Takahashi, J., Fukunaga, T., Yamakoshi, K.: DFA mechanism on the AES key schedule. In: Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 62–74 (2007)
54. Tunstall, M., Mukhopadhyay, D., Ali, S.: Differential fault analysis of the advanced encryption standard using a single fault. In: Proceedings of the 5th IFIP WG 11.2 International Conference on Information Security Theory and Practice: Security and Privacy of Mobile Devices in Wireless Communication (WISTP), pp. 224–233 (2011)
55. Wu, K., Karri, R., Kuznetsov, G., Goessel, M.: Low cost concurrent error detection for the advanced encryption standard. In: International Test Conference (ITC), pp. 1242–1248 (2004)
56. Guo, X., Mukhopadhyay, D., Jin, C., Karri, R.: Security analysis of concurrent error detection against differential fault analysis. J. Cryptogr. Eng. (2014)

Part II
Hardware Counterfeiting and Integrity Protection

Circuit Timing Signature (CTS) for Detection of Counterfeit Integrated Circuits

Kan Xiao, Domenic Forte, and Mohammad (Mark) Tehranipoor

Abstract Counterfeit integrated circuits (ICs) have been on the rise over the past decade and represent a major concern. Counterfeits impact the security and reliability of electronic systems particularly those deployed in critical applications. While there are several different types of counterfeit ICs in the supply chain (cloned, recycled, remarked, overproduced, etc.), reports indicate that recycled ICs constitute the majority of all counterfeit ICs in the market today. Such ICs are recovered from the scrapped boards of used devices. Since these ICs are identical to their unused counterparts in appearance, functionality, and packaging, detecting them can be challenging. It has been observed that path delays in recycled ICs will be larger than those in unused ICs due to the effects of silicon aging, such as negative/positive bias temperature instability (NBTI/PBTI) and hot carrier injection (HCI). In this chapter, a circuit timing signature (CTS) technique is presented to distinguish recycled ICs from unused ones. Specifically, a clock sweeping technique is employed both to measure the amount of path delay and to generate a timing signature for chips under testing. Due to the degradation in the field, the path delay distribution of recycled ICs becomes different from that of new/unused ICs, resulting in a different CTS. An authentication flow for accurately identifying recycled ICs is presented. Results show that statistical analysis can effectively separate the impact of process variations from aging effects on path delay. In addition, the CTS is extended to the detection of cloned ICs, overproduced ICs, and remarked ICs. A unique binary ID is generated based on the CTS of each IC. By checking the intrinsic IDs, cloned, overproduced, and remarked ICs can be effectively identified.

K. Xiao (✉)
ECE Department at University of Connecticut, Fairfield Way, Unit 4157,
Storrs, CT 06269-4157, USA
e-mail: kax10001@engr.uconn.edu

D. Forte • M. (Mark) Tehranipoor
Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA
e-mail: tehrani@engr.uconn.edu

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_6

1 Introduction

The counterfeiting of semiconductor components has been on the rise for many years as a result of several vulnerabilities in the electronics component supply chain [1]. The most recent data provided by Information Handing Service Inc. (IHS) shows that reports of counterfeit ICs have quadrupled since 2009 [2]. The Senate Armed Services public hearing on this issue and the subsequent report clearly identified counterfeit detection as a major issue to address [3, 4]. Counterfeit components are of great concern to government and industry because of their reliability issues and potential system failures or malfunctions that cause mission failures [5, 6].

According to the definition of counterfeit electronics [7], a counterfeit component (1) is an unauthorized copy; (2) does not conform to original component manufacturer (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. Based on the definitions above and analysis of supply chain vulnerabilities, Guin et al. [6, 8] classified the counterfeit types into seven distinct categories as shown in Fig. 1.

- **Recycled:** It refers to an electronic component that is reclaimed/recovered from a system and then modified to be misrepresented as a new component of an OCM. Recycled parts may exhibit lower performance and shorter lifetime due to aging phenomena from their usage. Further, the reclaiming process (removal under a very high temperature, aggressive physical removal from boards, washing, sanding, repackaging, etc.) could damage the part(s), introduce latent defects, or make them completely non-functional due to exposure to extreme conditions in an uncontrolled environment. Such parts will be unreliable and render the systems that incorporate them also unreliable.
- **Remarked:** Most legitimate components contain markings on their packages that indicate manufacturer, trademark, part number, grade, lot code, etc. The remarking is accomplished by either chemically or physically removing the original marking, blacktopping (resurfacing) the surface to hide any scratches or imperfections that have been created, and then remarking the new surface.

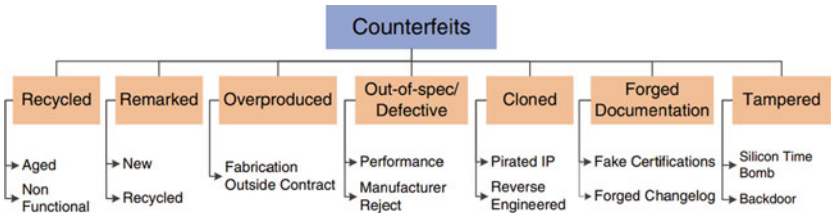


Fig. 1 A taxonomy of counterfeit component types

The primary incentive for remarking is to drive up a component's price on the open market or to make a dissimilar lot fraudulently appear homogeneous. For example, industrial or defense grade components are more valuable than commercial grade because of their superior durability and performance. However, remarked commercial grade components sold as military grade will not be able to withstand the harsh conditions of their more durable counterparts.

- **Overproduced:** Due to globalization, design houses outsource their designs for fabrication and packaging to companies all around the world, mainly to reduce the manufacturing cost. Overproduction occurs when foundries and packaging companies sell components outside of contract with the design house (component's intellectual property (IP) owner). Aside from the loss in profits for the IP owner, overproduced ICs may pose serious reliability threats since they are often not subjected to the same rigorous testing as authentic parts and may not meet the manufacturer's standard flow requirements.
- **Out-of-Spec/Defective:** A part is considered defective if it produces an incorrect response to post-manufacturing tests. These parts should be destroyed, downgraded, or otherwise properly disposed of. However, if they instead are sold on the open markets, either knowingly by an untrusted entity or by a third party who has stolen them, there will be an unknown increase in risk of failure.
- **Cloned:** Cloning is widely used by a range of adversaries/counterfeiters (from small entities to large organizations) to copy a design in order to eliminate the large development cost of a part. Cloning can be done in two ways by reverse engineering, and, by obtaining intellectual property (IP) illegally (also called IP theft). Cloning can also occur with unauthorized knowledge transfer from a person with access to the part design.
- **Forged Documentation:** The documentation shipped with a component contains information regarding specification, testing, Certificates of Conformance, Statement of Work, etc. By modifying or forging these documents, a component can be misrepresented and sold even if it is nonconforming or defective. It is often difficult to verify the authenticity of such documents because the archived information for older designs and older parts may not be available at the OCM. Legitimate documentation can also be copied and associated with parts from a lot not corresponding with the legitimate documentation.
- **Tampered:** Components that are tampered can have dangerous consequences for the systems that incorporate them. For example, tampered chips can act as silicon time bombs where their functionality is unexpectedly "killed" at a critical moment. Tampered chips may also contain backdoors that give access to critical system functionality or leak secret information to an adversary. A detailed taxonomy for tampering with a device at the die level (i.e., hardware Trojan) can be found in [9].

The category that has contributed the most to the rise of counterfeits is the recycled ICs. It is estimated that these recycled ICs account for 80 % of all counterfeits being sold worldwide. In addition, electronics consumer and e-waste trends suggest that this recycling is only going to increase over time as more gadgets are used for shorter periods of time [10, 11]. For example, US consumers threw away 90 million cell phones in 2007 and 135 millions cell phones in 2010 and only about 20 % were recycled properly [10]. The growth of this type of counterfeit is worrisome for two reasons: the reliability and security concerns that these recycled ICs present, and the difficulties involved in detecting them. In addition, recycled ICs may also have been further tampered with during the recycling process, and thus represent additional reliability and security risks [12].

Most existing techniques for counterfeit detection are limited and cannot cover all counterfeit types. Some recycled ICs may be detected through careful visual inspection since the markings or parts of the package may have been damaged during the refining process. However, most recycled ICs are refined by professional remarking, packaging, and cleanup processes, making it difficult to identify, since they have similar appearance and basic functionality as their fresh counterparts. Several approaches have been proposed to identify recycled ICs [13–16]. Zhang et al. [13] and Guin et al. [14] use a light-weight on-chip sensor that avoids the data collection altogether and applies a "self-referencing" concept to the measurement of use time. However, this approach cannot address detection of existing and legacy ICs that have no such sensors embedded in them. In addition, Silicon physical unclonable functions (PUFs) have been developed to generate unique identifiers for each IC based on process variations [17–19], which can be used to detect cloned, overproduced, and remarked ICs. Hardware metering is a set of security protocols that enables the design house to achieve post-fabrication control of the overproduced ICs [20, 21]. Secure Split Test (SST) secures the manufacturing test process to prevent counterfeits, such as cloned, overproduced, and defective/out-of-spec ICs [22]. However, these approaches do not address the issue of recycled ICs.

In this chapter, a circuit timing signature (CTS) technique for detection of recycled, remarked, overproduced, and cloned ICs is presented. A circuit timing signature (CTS) is generated for each IC by performing a clock sweeping technique. For fresh ICs, the delay distribution of paths will be within a certain range. The circuit timing signature of the fresh ICs can be generated during manufacturing test of these ICs and can then be stored in a database for future use when identifying recycled ICs. Due to aging effects, such as negative/positive bias temperature instability (NBTI/PBTI) and hot carrier injection (HCI), the path delays in recycled ICs will be larger than those in fresh ICs. For a chip under authentication (CUA), the larger the path delays are, the higher the probability is that the CUA has been used and is a recycled IC. In this chapter, a circuit timing signature (CTS) generation procedure using a clock sweeping technique is introduced. Statistical data analysis is used to distinguish the path delay changes caused by process variation from those path delay changes caused by aging.

This chapter also discusses how variants of the CTS technique is used to detect other counterfeit types, such as remarked, overproduced, and cloned ICs, so it can cover the majority of counterfeit types [23]. These counterfeits introduce new ICs which will have different characteristics due to different process technologies and/or variations. Since CTS can effectively reveal differences in timing characteristics, CTS is unique for each IC and can be employed to identify new ICs. The authentication flow for uniquely identifying ICs under measurement and environmental noise is also presented in this chapter. The circuit timing signature (CTS) created by the clock sweeping technique represents a novel improvement in existing ideas for several reasons. First, our technique can be applied to ICs already in production, including legacy designs. Second, it uses data that can be obtained through the use of existing pattern sets and testing hardware capabilities. Third, no additional hardware is necessary—there is no area, power, or timing overhead to the circuit. Finally, because the circuit is not added to or modified in any way, no attacks on the circuit are possible. Further, it is difficult if not impossible to reverse aging or alter process variations to modify the CTS signatures.

The rest of the chapter is organized as follows: Sect. 2 presents background on the impact of aging and process variations on different gates and paths. Section 3 discusses the clock sweeping technique and CTS generation procedure. The approaches for identifying recycled, remarked, overproduced and cloned ICs with statistical data analysis are presented in Sects. 4 and 5. Experimental results from simulation and FPGA implementation are presented in Sect. 6. Finally, a summary is given in Sect. 7.

2 Path Delay Analysis on Process Variation and Degradation

2.1 Path-Delay Degradation Analysis

When a chip is used in the field, aging effects cause some of its parameters to shift over time. This section will briefly discuss the mechanics of the two most common wearout mechanisms for a CMOS device: Negative Bias Temperature Instability (NBTI) and Hot Carrier Injection (HCI). Both mechanisms deteriorate transistor performance.

- NBTI occurs when traps are accumulated on the boundary of the Si-SiO_2 interface, which usually happens when PMOS is reversely biased. NBTI increases the absolute value of the PMOS threshold voltage and results in decreasing drain current, decreasing transconductance of a MOSFET and consequently increasing gate propagation delay [24, 25].
- HCI is another major wearout mechanism. Drain-source electrical field accelerates the charge carriers to reach a high kinetic energy, which enables the avalanche of secondary carriers through impact ionization. The process consequently creates traps at the gate dielectric/silicon substrate interface, as well as dielectric bulk traps, and therefore degrades device characteristics including voltage threshold [26, 27]. While HCI was considered less dominant than the NBTI in the past, continued semiconductor scaling has increased its impact significantly.

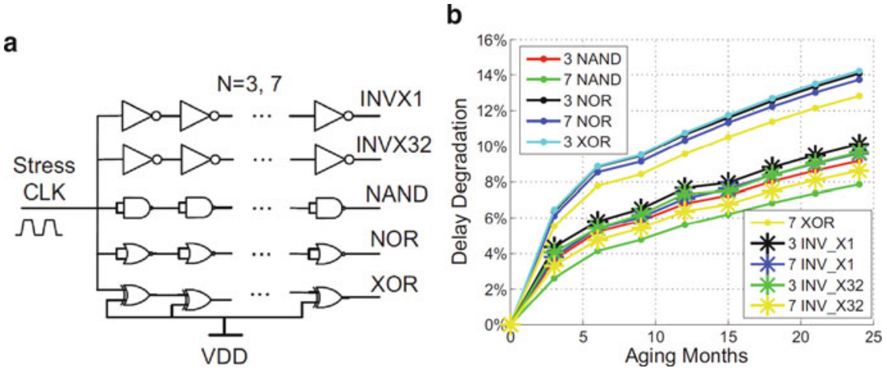


Fig. 2 (**a**) An illustrative circuit with NAND-gate, NOR-gate, XOR-gate, and inverter chains and (**b**) Delay degradation of the chains

Since recycled ICs have been impacted by all of these aging effects, the path delay of recycled ICs will be different from those of fresh ICs. In order to demonstrate the impact of aging on path delay in ICs, in a simple manner, different gate chains were simulated using 45 nm technology (http://www.nangate.com/) as shown in Fig. 2a. The simulation was conducted by HSPICE MOSRA [28] with the built-in aging model [28] and combined NBTI and HCI aging effects at a temperature of 25 °C. Standard threshold voltage (SVT) INVX1, INVX32, NAND, NOR, and XOR gate chains of different lengths were simulated for up to 2 years of usage. Figure 2a shows that all chains are experiencing stress from a 500 MHz clock. Any other stress (e.g., DC stress which is a constant “0” or “1”, or AC stress with different duty ratios) and usage time could be used in this simulation. Figure 2b presents the delay degradation caused by 2 years (24 months) of aging. It is clearly evident that different gate chains age at slightly different rates, which depends on the structure of the gates. The XOR gate chain has the fastest aging rate amongst these chains. Comparing the delay degradation rates of the INVX1 and INVX32 chains, it can be concluded that larger gates will age at a lower rate than smaller gates under the same stress. In addition, the workload (input value and the switching frequency of each gate) also has a significant impact on the aging rate. ICs may be recycled from different used boards from different users who may have applied different workloads to the IC at different times. Therefore, it is practically impossible to know the exact input vectors applied by the user. The impact of workload on a chip’s path delay degradation will be discussed in detail in Sect. 4.1.

Figure 3a shows the delay of a randomly selected critical path P_i (this path includes 22 gates) from the ISCAS’89 benchmark s38417 with stress from a random workload (functional patterns) applied to the primary inputs. The path was aged for 4 years with NBTI and HCI effects at room temperature 25 °C. The simulation results indicate that the degradation of path P_i after 1 year is around 10 % while if the circuit is used for 4 years, the degradation is about 17 %, indicating that most aging occurs at the early usage phase of the design. Therefore, if there are

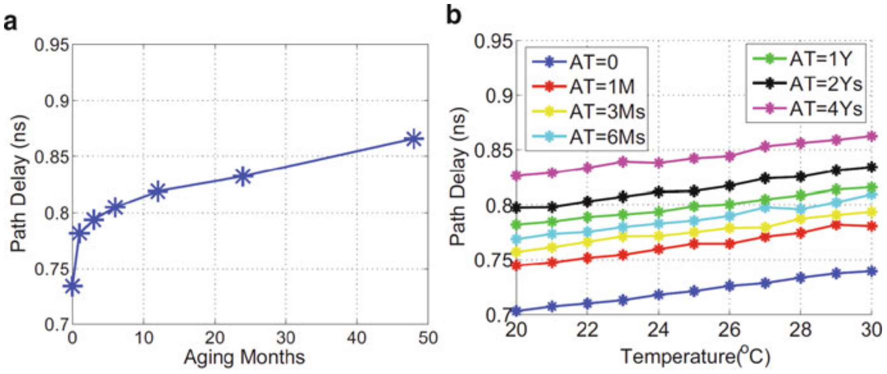


Fig. 3 (**a**) Delay degradation of path P_i and (**b**) P_i delay increases with increased temperature

no environmental or process variations, such degradation should provide great a opportunity to identify recycled ICs by measuring one path delay from the circuit. However, these variations have a significant impact on the path delay. On the other hand, different paths age at different rates as demonstrated earlier in this section. Figure 3b shows the delay of path P_i under different temperatures at different aging times. In the figure, AT denotes aging time where M and Y denotes months and years respectively. This plot shows that the delay of path P_i increases as the temperature goes up and paths age with a faster speed at a higher temperature.

2.2 Path Delay and Process Variation

In theory, the specifications and functionality of different ICs of the same design should be identical. In practice, this is not the case due to uncontrollable manufacturing variations that limit our ability to accurately create IC structures at smaller technology nodes. Thus, values such as the threshold voltage of a transistor (V_{th}), the gate length of the transistor (L), and the oxide thickness of the transistor (T_{ox}) can only be guaranteed to be within some ranges. Variation on these and other parameters is important to take into consideration because these parameters directly affect device performance. For example, the delay of a traditional CMOS inverter can be expressed in two Eqs. (1) and (2) [29], where the high-to-low and low-to-high propagation delays t_{PHL} and t_{PLH} are affected by variation in the three previously mentioned parameters, as shown in (3) and (4).

$$t_{pHL} = ln(2)R_{eqn}C_L \tag{1}$$

$$t_{pLH} = ln(2)R_{eqp}C_L \tag{2}$$

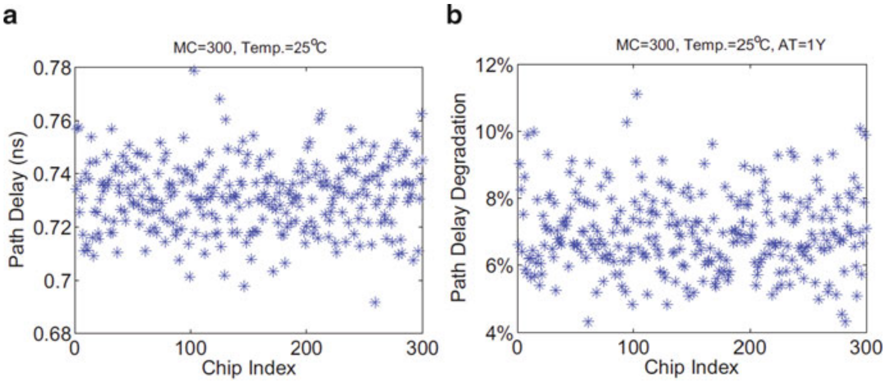


Fig. 4 (**a**) Delay degradation of path P_i and (**b**) P_i delay increases with increased temperature

with

$$R_{eq} = \frac{1}{V_{DD}/2} \int_{V_{DD}/2}^{V_{DD}} \frac{V}{I_{DSAT}(1+\lambda V)} dV \approx \frac{3}{4} \frac{V_{DD}}{I_{DSAT}} (1 - \frac{7}{9} \lambda V_{DD}) \tag{3}$$

and

$$I_{DSAT} = k' \frac{W}{L} ((V_{DD} - V_{th}) V_{DSAT} - V_{DSAT}^2) \tag{4}$$

To analyze the impact of process variations on a path P_i's delay, Monte Carlo simulation were performed using HSPICE on s38417. 300 Monte Carlo simulation results of P_i at 25 °C are shown in Fig. 4a, with 3-sigma 2 % T_{ox}, 5 %V_{th}, and 5 % L inter-die and 1 % T_{ox}, 5 %V_{th}, and 5 % L intra-die process variations. It shows that P_i's delay varies around 12 % due to process variations. In addition, process variations also have a significant impact on the aging rate of the path delay, as shown in Fig. 4b. P_i's delay degradation in the 300 ICs varied around 8 % (4∼12 %) for 1 year of aging. *These variations evidently make the detection difficult since the path delay shifts caused by aging effects in recycled ICs must be separated from those caused by process variations in fresh ICs.*

3 Clock Sweeping Technique for Circuit Timing Signature Generation

For a manufactured chip, it is very difficult to measure the delay of a particular path. Conventional VLSI delay tests have been developed to identify timing violations introduced by manufacturing defects that result in slow-to-fall or slow-to-rise signals. Basically, delay tests (using path delay fault (PDF) patterns and transition delay fault (TDF) patterns) just check whether critical paths meet timing constraints but do not measure the actual path delay [30]. In addition, conventional delay tests focus on critical paths that are long and have a higher probability of resulting in timing violations, but other shorter paths are not covered in conventional VLSI delay tests. Therefore, several techniques have been proposed to measure path delay in manufactured chips. Datta et al. [31] measure path delay using a Modified Vernier Delay Line (MVDL), which converts path delay into a digital "0" and "1" series in flip-flops forming MVDL and improves the measurement resolution below 100 ps. Ghosh et al. [32, 33] utilize built-in delay sensors, which use of saw-tooth voltage generators and comparators to convert path delays from the analog domain to the digital domain. Su et al. use phase detectors to detect phase differences between input and output, and then translate it to bus delays [34]. Wang et al. [35] further improved on-chip delay measurement architectures and developed an oscillator from a targeted path for which it is used to measure path delay on-chip under the impact of process variations. To alleviate accuracy degradation caused by the architecture itself, a high-accuracy calibration process is presented as well. However, these techniques introduce large overheads or high costs. Each approach requires additional circuitry on the target paths for delay measurements. As the number of paths increases, the silicon overhead and yields are heavily impacted. Moreover, the test time is increased if each path is tested separately.

Our approach relies on "clock sweeping" which does not suffer from the above-listed issues. Clock sweeping is a common practice in industry and is used for the speed binning of parts where test patterns are applied at different clock frequencies, from a lower speed to higher speeds. Some paths that are sensitized by the delay test pattern which are longer than the current clock period, start to fail when the clock speed increases. The obtained start-to-fail clock frequency can indicate the delays of the paths sensitized by these patterns. The clock sweeping technique can target multiple paths without any design or silicon overhead.

The clock sweeping is illustrated in the following example. Assume that the six paths in Fig. 5a can be sensitized by test patterns. The clock period is swept from f_0 to f_5 (suppose f_0 is circuit functional frequency and f_5 is the maximum allowable

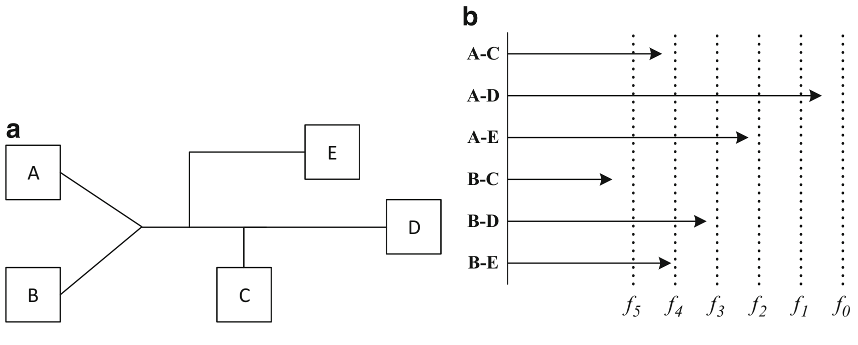


Fig. 5 (**a**) An example circuit and (**b**) Clock sweeping on paths in the circuit

frequency) and the sweep step size is Δt ($\Delta f = f_i - f_{i+1}$, i = 0, 1, 2, 3, 4), as shown in Figure 5b. The length of each arrow denotes the path delay of each path. As an example, path B-D is able to propagate correct values at frequency f_0 to f_3 (pass), and will produce incorrect logic values at frequency f_4 (fail). Thus, its start-to-fail clock frequency is f_4, which denotes the length of path B-D is between the frequency f_3 and f_4. When the clock is swept from low frequencies to high frequencies, paths will fail sequentially, with longer paths failing before shorter paths. If the sweep range is large enough, many paths will fail during clock sweeping, allowing their delay information to be obtained, even for hazardous patterns if the size of the hazard is larger than the sweep step size.

The ability to perform clock sweeping on a path is limited by the availability of sweeping clock frequencies, the degree to which we can excite paths in the circuit, and the lengths of the paths in the IC. The range and step size of sweeping frequencies are two critical parameters involved in the clock sweeping technique. A smaller step size could be more sensitive to small delay changes caused by the wearout mechanism and process variation. The range determines the range of testable paths (long paths). Higher maximum frequency could fail more long paths during clock sweeping. However, the maximum frequency applied to the circuit cannot go very high, because it is also restricted by the design characteristics, path-delay distribution in the design, the maximum power consumption, and the on-chip or off-chip frequency generator's limit. For example, with the Ocelot ZFP tester [36], the main frequency is 400 MHz and the frequency step size is 1 MHz.

Figure 6 shows the flow of the clock sweeping technique. The delay test patterns are applied to ICs at different clock frequencies. By analyzing the pass and fail values with clock sweeping, the start-to-fail frequency at each flip-flop for each pattern can be obtained. As an example, Table 1 presents the start-to-fail frequency for each pattern/flip-flop combination in ICs. Patterns are able to sensitize different kinds of paths at different flip-flops: long, short or no path at all. For these flip-flops

Fig. 6 Clock sweeping flow

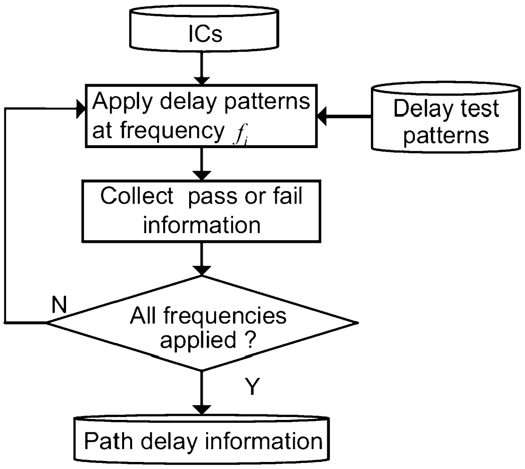

ICs
Apply delay patterns at frequency f_i
Delay test patterns
Collect pass or fail information
All frequencies applied ?
N
Y
Path delay information

Table 1 Pattern/flip-flop (Pi/FFj) combinations with start-to-fail frequencies.

	P1/FF1	P1/FF2	...	P2/FF1	P2/FF3	...	PP_1/FFm
IC 1	f_6	f_{14}	...	f_3	f_{20}	...	f_{10}
IC 2	f_7	f_{12}	...	f_4	f_{21}	...	f_9
IC 3	f_5	f_{11}	...	f_3	f_{19}	...	f_{10}
...	...	...	...	...	...	...	...
IC n	f_8	f_{14}	...	f_4	f_{23}	...	f_{12}

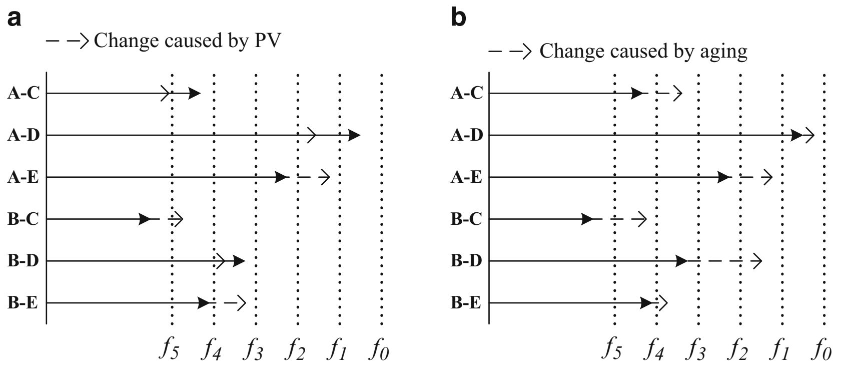


Fig. 7 Clock sweeping on paths in the example circuit if there is (**a**) process variations or (**b**) degradation due to aging

at which short or no paths are sensitized (like the path B-C in Figure 5b), they always capture "passing" values during the clock sweeping. These invalid pattern/flip-flop combinations need to be discarded. For example, assuming the pattern 2/flip-flop 2 (P2/FF2) is an invalid combination, the column P2/FF2 has been removed from Table 1. The remaining valid elements form *circuit timing signature* (CTS), as is shown in Table 1. Each row of the table is a CTS for an IC and each column is an element of the CTS.

Figure 7a shows examples of paths in two fresh chips. Although they have the same design, the path delay varies because of process variations. Assuming all these six paths in the figure can be tested by test patterns, so the CTS using the combination of these six paths from top to bottom in two chips, are "$f_5\, f_1\, f_3 X\, f_4\, f_5$" and "$X\, f_2\, f_2\, f_5\, f_4\, f_4$", respectively (X represents no start-to-fail frequency is found during clock sweeping). If these two chips have been used in the field and experience various stresses, the aged paths are shown in Figure 7b. The degradation due to aging effects will result in a small extra delay on an aged path, which may push the arrow to the right, so the CTS change to "$f_4\, f_1\, f_2\, f_5\, f_2\, f_4$" from "$f_5\, f_1\, f_3 X\, f_4\, f_5$". Therefore, process variation and aging degradation could be revealed in the CTS using the clock sweeping technique. The next two sections will exploit the aging to identify recycled ICs and the process variations to identify the cloned, overproduced, and remarked ICs.

4 CTS for Detecting Recycled ICs

As described in Sect. 3, the clock sweeping technique can generate a CTS for each chip. Figure 8 shows the flow for identifying recycled ICs using circuit timing signature (CTS) and statistical analysis. The flow is divided into three major steps. First, paths are simulated and selected according to their aging rate. Next, the delay information of these paths is measured by a clock sweeping technique in fresh ICs (either during manufacturing test on all ICs or during authentication on a sample of fresh ICs) and in any chip under authentication (CUA). Finally, statistical analysis is used to decide whether the CUAs are recycled ICs or not.

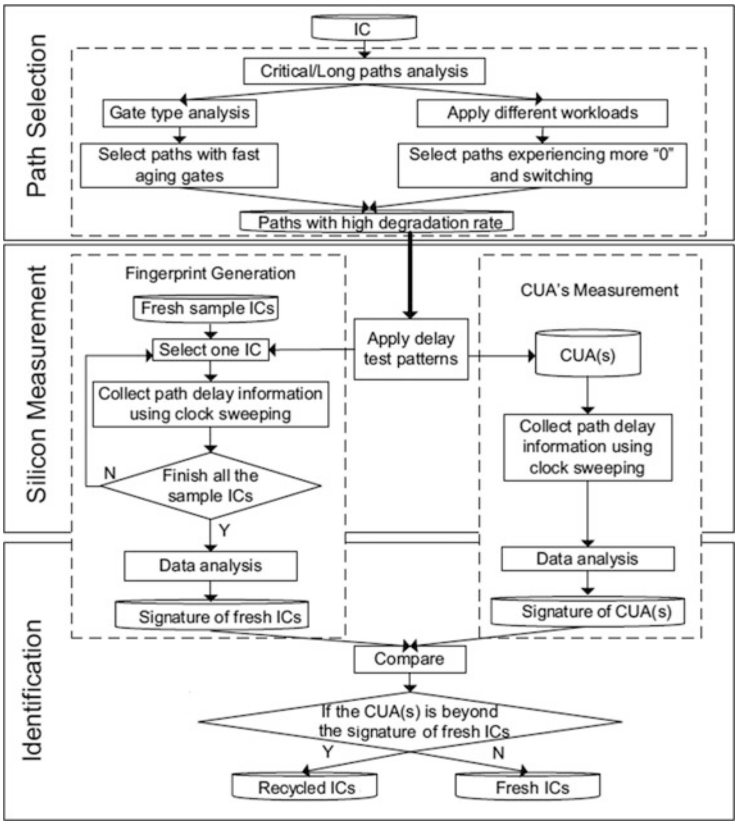

Path Selection
IC
Critical/Long paths analysis
Gate type analysis
Apply different workloads
Select paths with fast aging gates
Select paths experiencing more "0" and switching
Paths with high degradation rate
Silicon Measurement
Fingerprint Generation
Fresh sample ICs
Select one IC
Collect path delay information using clock sweeping
Finish all the sample ICs
N
Y
Apply delay test patterns
CUA's Measurement
CUA(s)
Collect path delay information using clock sweeping
Identification
Data analysis
Signature of fresh ICs
Data analysis
Signature of CUA(s)
Compare
If the CUA(s) is beyond the signature of fresh ICs
Y
N
Recycled ICs
Fresh ICs

Fig. 8 Recycled IC identification flow

4.1 Circuit Timing Signature Generation Considering Aging

4.1.1 Step 1: Path Selection

Due to the large number of critical and long paths in a circuit, the first step is to select paths which age at faster rates by analyzing the gate types in different paths and simulating the circuit with different workloads [37, 38]. Paths with higher rates of aging are preferred for CTS generation, since the differences in the delay of those paths between recycled ICs and fresh ICs will be much larger than the differences in paths, which degrade more slowly. CTS generated by fast-aging paths could help identify recycled ICs used for a shorter time. However, there are several parameters impacting the aging rate of a path, including the type of gates composing the path and the workload (as discussed previously in Sect. 2.1). Based on these parameters and the observations made from the simulation shown in Fig. 3, the following rules are developed to select the paths that age at the fastest rates: (1) paths with more fast-aging gates, such as NOR or XOR gates, will be selected, and (2) paths that experience more zeroes and more switching activity will be selected. More zeroes in the path will increase the effect of NBTI on the PMOS transistors, and a high switching frequency will increase the HCI effects on gates, increasing the path delay degradation more significantly.

Paths with more fast-aging gates would be identified by analyzing the type of gates composing the paths. However, it is very difficult to identify paths that experience more zeroes and more switching activity without knowing the specific workload. Therefore, in this work, different random workloads (input combinations) are applied to ICs' primary inputs during logic simulation. For each gate on a critical path, the average switching activity and the zeroes it has experienced are calculated. Paths with more switching activity and zeroes are then selected using our flow. These paths, along with those composed of the more fast-aging gates, are used to generate CTS signatures to identify recycled ICs. The number of selected paths could be adjusted according to the design, its testing procedure, and the desired detection confidence. In our simulations (see Sect. 6.1.1), the top 50 paths with fast-aging gates are selected and the top 50 paths experiencing more switching activity and zeroes in the benchmark circuit.

4.1.2 Silicon Measurement

The second step in Figure 8 is to collect the selected paths' delay from the ICs. Note that the signature generation can be done during a manufacturing test of a large sample of ICs before shipping them to the market or on a number of fresh ICs from each production kept by the design house for the purpose of authentication or recycled ICs identification. The larger the size of the sample is, the wider the range of process variations that will be included in the signature, reducing the probability of wrongly identifying fresh ICs with large process variations as recycled ICs. Path delay information from the fresh ICs is measured by performing test procedures on the ICs. Traditionally, test patterns are generated by ATPG before fabrication to detect path and transition delay faults. These patterns will be applied to all fresh ICs using clock sweeping techniques, as described in Sect. 3. A CTS will be generated for each IC as shown in Table 1.

4.1.3 Identification

Once the path delay in all sample chips is measured, statistical data analysis will be used to generate a signature for fresh ICs. The more the sample chips are, the more process variations will be covered, reducing the probability that fresh ICs with large process variations will be identified as recycled ICs. For a circuit under authentication (CUA) taken from the market, the same test patterns will be applied in a controlled environment (near-identical to that used for the fresh chips). The path delay information of the CUA will be processed by the same statistical data analysis methods. In a simple analysis, if the signature of the CUA is outside of the range of the fresh ICs' signature, there is a high probability that the CUA is a recycled IC. Otherwise, the CUA is likely a fresh IC. The longer the CUA has been used, the more aging effects it will have experienced, making it easier to identify.

Without extra hardware circuitry embedded into the ICs, our recycled IC identification technique imposes no area or power overheads. It provides a negligible test time overhead during manufacturing test on a sample of ICs, since only a few patterns must be applied several times at different frequencies. Also, there is no change in the current IC design and test flow since there is no additional circuitry in the IC used for detection. In addition, this method is resilient to tampering attacks. It is inherently difficult for recyclers to mask the impact of aging on the recycled ICs' path-delay signatures during the recycling process. The only limitations of this technique are that the design must be known to get the path delay information and signatures from fresh chips are required to compare with the CUAs for classification.

4.2 Statistical Data Analysis

Two statistical data analysis methods are used in this paper: simple outlier analysis (SOA) and principal component analysis (PCA) [39]. When performing SOA, the flow randomly selects a single path from the selected path set, and uses its delay range in fresh ICs to generate a signature. The process variations of the CUA may or may not be the same as those within the sample ICs. The selected path delay of the CUA and sample ICs will follow the same distribution, which makes SOA effective in certain conditions. However, a single-path based analysis will not be very effective, due to the limited aging information collected. In general, this method is expected to be effective in distinguishing recycled ICs used for a long time from fresh ICs with small process variations, as demonstrated by our results in Sect. 6.

This basic approach is limited by the path that has been selected. One way we could improve this would be to use SOA independently on multiple paths and then make a final decision (recycled or fresh) according to a majority vote.

PCA is a dimensionality reduction technique that transforms the data into a reduced number of dimensions that captures most of the variations in the data and the PCA has been effective for detection of hardware Trojans in prior work [40, 41]. The PCA is applied to this problem as follows: The path delay information of all selected paths, which have been measured by clock sweeping, will be processed by PCA. In the simulations, the top 100 paths with faster aging rates were selected to generate signatures. The delay of each path is one of the variables for PCA to use. Therefore, with N ICs, the dimension of the data set for PCA to generate signature is $N \times 100$. We apply PCA and then plot the first three principal components (the three that capture the most variance). A convex hull is created around this data to represent the signature for fresh chips. The path delay information of the CUA was also analyzed by the same process and plotted in the same figure. If the CUA is outside of the convex created by the fresh ICs, there is a high probability that the CUA is a recycled IC.

5 CTS for Detecting Cloned, Overproduced, and Remarked ICs

A methodology to create a unique binary identity (ID) for each IC using CTS will be introduced in this section. The unique IDs are generated from intrinsic timing characteristics and are difficult to pirate and clone, so they can be used to address other counterfeit types that introduce new ICs, such as cloned, overproduced, and remarked. The procedure for uniquely identifying ICs with IDs can be broken down into two mains parts: (a) IC enrollment and (b) IC identification.

5.1 IC Enrollment

When the clock sweeping tests presented in Sect. 3 have been applied in the manufacturing test stage, CTS is obtained and can be utilized to generate a binary ID for each IC. Three steps are performed to enroll every manufactured IC:

1. Stability checking: a path selection procedure that selects the most stable measured paths from each IC
2. ID generation: a path delay comparison procedure which generates an unoptimized ID for each IC

3. ID optimization: an analysis and optimization procedure which increases the randomness and decreases the size of the IDs. The optimized ID will be stored in database for later access and authentication.

More details concerning the above steps are discussed in the forthcoming subsections.

5.1.1 Stability Checking

CTS is used to generate IDs, but path delay measurements using clock sweeping for CTS generation may be adversely affected by measurement and environmental noise. To improve the reliability of CTS, preprocessing is necessary to identify and discard unstable paths early on. Measuring an IC multiple times gives us multiple measurements of each path in that IC. Analysis of multiple measurements of a path can uncover whether that path is stable-it reported the same delay during each measurement—or unstable—it reported different delays on different occasions. To select stable paths, some subset of the IC set are measured multiple times and the m most stable paths in CTS are selected to be used to for further analysis on all n ICs.

5.1.2 ID Generation

Once the m most stable paths have been selected, the flow begins to generate IDs for each IC by performing comparisons between the delays of different paths from the ICs. Because of manufacturing variations, the path delay is somewhat uncertain from IC to IC, so that the result the comparison generated is random and unique. First, by sorting a list of m paths from an IC in ascending or descending order, a list of similar paths can be obtained. This sorting is called the "golden ranking", and this ordering is applied to the m paths of all n ICs. Once the golden ranking has been obtained, the path delay data for every IC is sorted by this ordering. For each IC, the process will traverse this ordered list of delays one element at a time. Whenever a delay in the list is greater than the next delay in the list, it will append a '1' onto the ID for that IC. Whenever a delay in the list is less than the next delay, it will append a '0' onto the ID for that IC. If m paths have been analyzed for each of the n chips, it will end up with n IDs of $m - 1$ bits each. The first/golden IC, which is used to determine the ordering, will have an ID of all ones or zeros, depending on whether the sort was ascending or descending.

5.1.3 ID Optimization

In order to improve the uniqueness of each ID, several optimization techniques have been developed. Some paths that are compared in the ID Generation step may produce the same value for all ICs because the path delays are very different.

This will reduce the average Hamming distance between IDs as this bit will be the same in all ICs. If enough paths have been measured, it may be possible for an IC designer to pick and choose which comparisons they are going to perform to create their IDs. The comparisons can be chosen by analyzing each bit of the $m - 1$ bit "unoptimized IDs", and deciding which bits have the least bias across all IDs. A bit should be included in the "optimized" ID if the number of zeros in that bit position is close to 50 % of the number of IDs-if the number of zeros in that bit position is in the range of $\frac{n}{2} \pm \varepsilon$, where ε is some small constant. This process selects the most unpredictable comparisons across the IC data set. By performing this optimization technique, IDs become shorter and more random, at the cost of discarding some of the data obtained during the measurement process.

5.2 IC Identification

This IC enrollment process will be run on every IC during manufacturing tests. The IDs for every manufactured IC will be stored in a database for later access. Parts of enrollment process will be run again later when engineers wish to identify an IC, and this is called IC the identification process. Figure 9 shows these two scenarios.

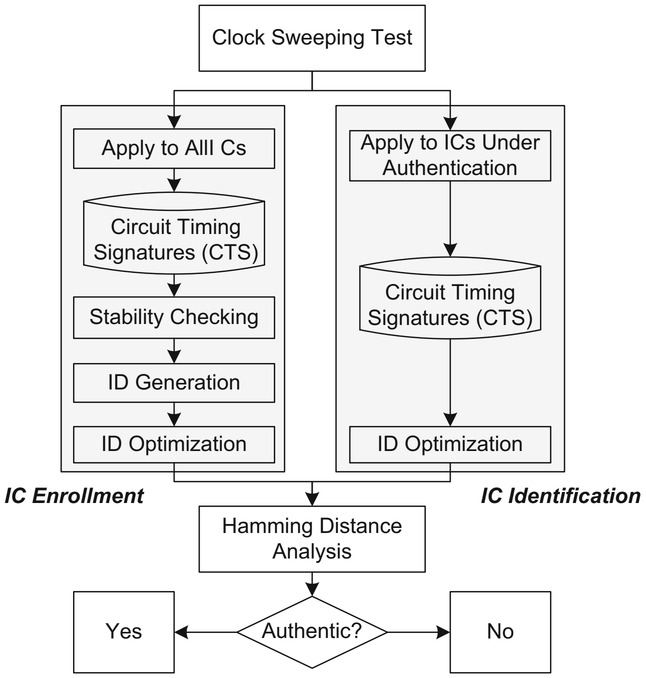


Fig. 9 Test generation and authentication flow

In order to identify an IC from the market, referred to as an IC under authentication (IUA) in Figure 9, test engineers will need: (1) the ID generated from reapplying the enrollment steps, and (2) the ID generated by the device during the original enrollment. Once they have the ID for the IUA, they can check to see if it is in the database using Hamming distance analysis. Hamming distance analysis is done by comparing the ID from the IUA to every ID in the database. If the ID from the IUA is an exact match to one of the IDs in the database, then they have identified the IUA.

Noise in the measurement process might make it so that there are a small number of bits in the ID which are different between the IC's ID in the database and its ID in the field. This issue has been addressed through the stability checking technique described in Sect. 5.1.1, but differences may still be present. Therefore, we define a *Hamming distance threshold*, based on analysis of the IDs during manufacturing tests. This threshold could be based off the minimum Hamming distance between two different ICs in the manufactured set. The threshold could also be based off of the maximum Hamming distance between IDs from the same IC, which would be created during the stability checking process. Exact thresholds would be derived from analysis of the ID set and measurement noise rates.

5.3 Overhead Analysis

This procedure incurs no area, power, or timing overhead. However, there are tests and computational overheads required to perform the enrollment steps. Each IC must undergo an additional k rounds of TDF testing at k different frequencies. If there are n ICs with P_{f_1} TDF patterns and N_{sff} flip-flops that will be tested at k frequencies, the worst case increase in testing time would be $n \cdot (k-1) \cdot P_{f_1} \cdot N_{sff} + N_{sff}$ clock cycles. This is a worst-case scenario as not all P_{f_1} TDF patterns will need to be tested at all k frequencies, as shown in Figure 3a. The stability checking procedure requires test engineers to measure a subset of $n_r \ll n$ ICs j times each, so there is a testing overhead of $n_r \cdot j \cdot (k-1) \cdot P_{f1} \cdot N_{sff} + N_{sff}$ clock cycles in the worst-case scenario. Judging the relative stabilities of each path in the ICs requires test engineers to individually analyze each of the m paths that were measured j times each in n_r ICs, resulting in a complexity of $O(n_r mj)$ time; however, n_r is much less than n, and j is going to be on the order of dozens to hundreds. The ID generation procedure requires test engineers to individually analyze each of the m paths in each of the n ICs, resulting in an $O(nm)$ runtime. The ID optimization technique analyzes each of the $m-1$ bits across all of the n IDs for a general runtime of $O(nm)$ as well. In general, m will be much smaller than n or n_r.

6 Experiment Results and Analysis

6.1 Analysis for Detection of Recycled ICs

The recycled IC identification flow and data analysis methods were implemented on several benchmarks in 45 nm technology. HSPICE MOSRA [28] was used to simulate the effects of aging on the path delay of different benchmarks. The supply voltage of the 45 nm technology is 1.1 V. Random workloads (random functional input patterns) were applied to several ISCAS'89 benchmarks. The circuit timing signature (CTS) was generated using clock sweeping at different aging times. Different process and temperature variations were also simulated to analyze their impact on the effectiveness of our recycled IC identification methods.

6.1.1 Process and Temperature Variations Analysis

Table 2 shows the three process variations rates that were used in simulations. Moving from PV0 to PV2, both inter-die and intra-die variations become larger. PV1 represents a realistic rate of process variations that a foundry might have. Four sets of Monte Carlo simulation (MCS) were run using different levels of variations, as shown in Table 3. For each set of MCS, 300 Monte Carlo simulations were run to generate 300 chips. During the simulations, the aging effects of NBTI and HCI were simulated with random stress for the benchmark s38417. From the top 500 paths, the paths $P_1, P_2, \ldots, P_{50}$ with fast-aging gates and the paths $P_{51}, P_{52}, \ldots, P_{100}$ with more zeros and higher switching activities were selected to generate CTS as described in Sect. 3.

Analysis Using Simple Outlier Analysis (SOA) First, 300 Monte Carlo simulations were run in MCS1. The maximum aging time is 2 years. Here, SOA was used to process the CTS. Three paths (P_1, P_2, and P_{51}) were selected to show the results of SOA. Figure 10a–c show the path delay distribution of the three paths from 300

Table 2 Process variation rates

	Inter-die (3σ)			Inter-die (3σ)		
	V_{th}	L	T_{ox}	V_{th}	L	T_{ox}
PV0	3 %	3 %	2 %	2 %	2 %	1 %
PV1	5 %	5 %	2 %	5 %	5 %	1 %
PV2	8 %	8 %	2 %	7 %	7 %	2 %

Table 3 Simulation setup

Experiments	Process variations	Temperature
MCS1	PV0	25 °C
MCS2	PV1	25 °C
MCS3	PV2	25 °C
MCS4	PV1	25 °C ± 10 °C

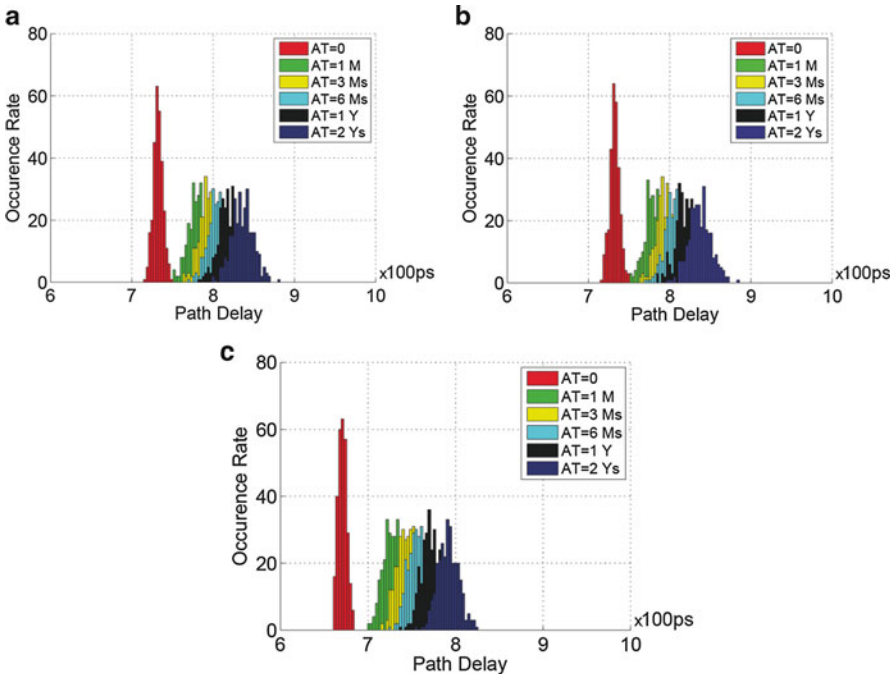


Fig. 10 Path delay distribution in ICs with PV0 in MCS1 at different aging times (**a**) Path P_1, (**b**) Path P_2, and (**c**) Path P_{51}

ICs used for different aging times. Similar results were obtained for the other 97 paths as well. For each path, the range of the path delay at AT='0' is the signature of the fresh ICs. Note that "M" and "Y" denote months and years of aging respectively. If the path delay of the CUA is out of the range specified at AT='0', there is a high probability that IC is a recycled one. Note the 300 different Monte Carlo simulations are used for recycled ICs from those used as sample fresh ICs. Figure 10 shows that the delay distribution of each path in recycled ICs shifts to the right, relative to the distribution of delays in fresh ICs. This is because the path delay in recycled ICs increases due to aging. The longer the ICs have been used, the more path delay degradation they will have experienced. In addition, Figure 10 also demonstrates that the path delay variation increases as the aging time increases. The reason for this is that ICs with different process variations age at different speeds, and the path delay variations become larger as the aging time increases.

Figure 10a shows the distribution of path P_1's delay. In the figure, the smallest delay of P_1 in recycled ICs used for 1 month is smaller than the largest delay in fresh ICs. Therefore, the detection rate of recycled ICs used for 1 month is less than 100 % (98.3 %) when the signature generated by SOA from path P_1 is used. However, the detection rate of recycled ICs used for 3 months or longer is 100 %, which demonstrates that it is easier to detect recycled ICs that have been used for

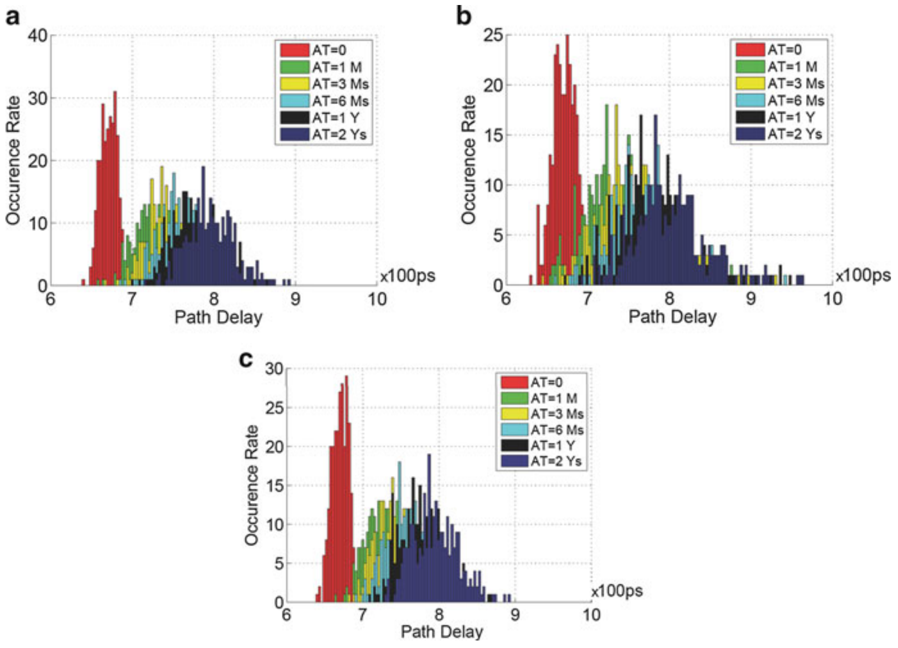


Fig. 11 Path P_{51} delay distribution in ICs at different aging times (**a**) in MCS2, (**b**) in MCS3, and (**c**) in MCS4

longer amounts of time. If the path P_2 is chosen to detect recycled ICs, the detection rate of ICs used for 1 month (95.7 %) is slightly less than when using path P_1. However, if path P_{51} is used, which has the fastest aging rate among the 100 paths, the detection rate is 100 % even if the ICs are only used for 1 month. P_{51} is the most effective path for identifying recycled ICs in this benchmark. From the above analysis, a conclusion is that different paths generate different signatures due to their different aging speeds, which makes SOA slightly less effective.

Figure 11a and b show the delay distribution of path P_{51} across 300 Monte Carlo simulations in MCS2 and MCS3. Overall, Figures 10c, 11a, and 11b present the delay distribution of the same path (P_{51}) in ICs with different process variations. By comparing these figures, it clearly shows that the larger the process variations are, the larger the path delay variations in fresh ICs will be, which makes it more difficult to detect recycled ICs. Even when using the most effective path P_{51}, the detection rates of ICs used for 1 month with PV1 and PV2 drop from 100 % with PV0 to 78.0 % and 50.7 %, respectively. A 100 % detection rate could be achieved if the ICs were used for 1 year or longer with PV1, or longer than 2 years with PV2.

300 Monte Carlo simulations were also run with ±10 °C temperature variation during the aging process in MCS4 as shown in Fig. 11c. The measurement temperature is 25 °C. It shows the delay distribution of path P_{51} and the detection rate of ICs used for 1 month using it is 67.7 %. Comparing Fig. 11a and c, the result presents

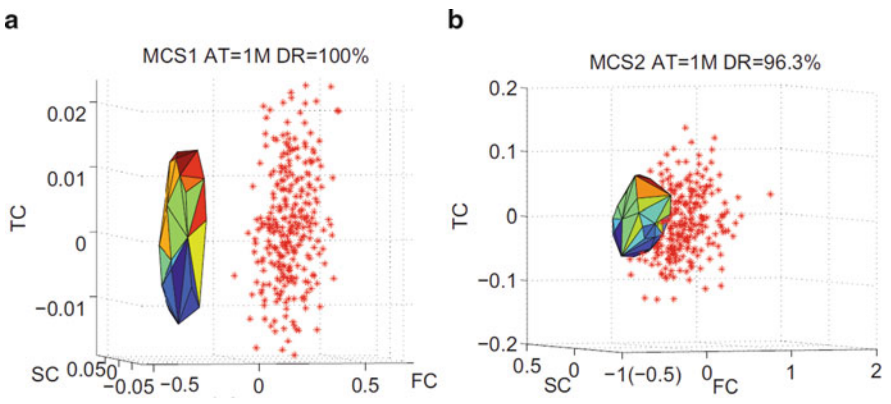


Fig. 12 PCA results of ICs under 25 °C (**a**) used for 1 month with PV0 in MCS1, (**b**) used for 1 month with PV1 in MCS2, and (**c**) used for 3 months with PV1 in MCS2

that the larger the temperature variation is, the larger the path delay variation is, which makes it even more difficult to detect recycled ICs.

Analysis Using Principal Component Analysis (PCA) A similar analysis is done using PCA for different MCSs. Figure 12a shows the PCA results of the 100 paths in s38417 with 300 chips in MCS1. "FC" denotes the first component from PCA, "SC" represents the second component, "TC" is the third component, and "DR" denotes the detection rate. The convex hull boundary is built up from fresh IC data, and represents the signature for fresh ICs. The red asterisks represent chips used for 1 month. In the figure, the 300 used ICs were completely separated from the signature of the fresh ICs. Thus, the detection rate using timing signatures generated by PCA is 100 % for recycled ICs used for *1 month*. For recycled ICs used for a longer time, the detection rate will obviously be 100 % as well.

The path delay information from the remaining three sets of MCSs was also analyzed by PCA. Figure 12b shows the analysis results of fresh chips and recycled ICs used for 1 month in MCS2. The three-dimensional figure shows that some of the recycled ICs are close to the fresh ICs' signature. The detection rate is 96.3 %, which is much higher than using SOA. Comparing Figure 12b and a, it shows that (1) the convex hull built up from fresh ICs in MCS2 is much larger than that in MCS1 (note that the convex hull in MCS1 looks larger than MCS2 due to its small scale of axes), and (2) the recycled ICs in MCS2 are closer to fresh ICs than those in MCS1, which makes the detection rate in MCS2 less than that in MCS1. The path delay information of 300 ICs used for 3 months in MCS2 were also processed, and the results are shown in Figure 12c. Comparing Figure 12b and c, it shows that the longer the chips have been used, the farther they will be from the fresh ICs'

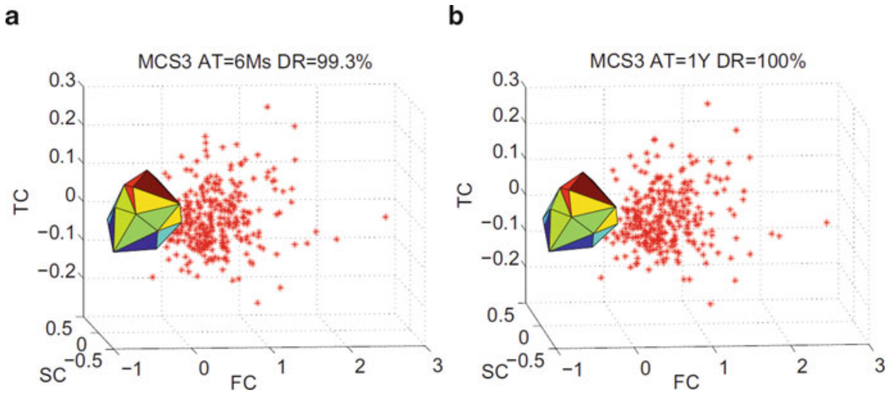


Fig. 13 PCA results of ICs with PV2 under 25 °C in MCS3 used for (**a**) 6 months and (**b**) 1 year

signature. The detection rate of recycled ICs used for *3 months or longer* with PV1 at 25 °C is 100 %.

Figure 13 are the PCA results of ICs in MCS3. The detection rate of recycled ICs used for 1 month, 3 months, 6 months, and 1 year are 72.7 %, 89.3 %, 99.3 %, and 100 %, respectively. The figures of PCA results of recycled ICs used for 1 month and 3 months are not shown here since the detection rates are so far from 100 %. Figure 13a and b show the fresh ICs' signatures and the recycled ICs used for 6 months and 1 year, respectively. The recycled ICs used for longer times are easier to detect, as seen by comparing Fig. 13a and b. The detection rates in these simulations, prove that it is more difficult to detect recycled ICs that have higher levels of process variations. The 99.3 % detection rate of ICs used for 6 months and the 100 % detection rate of ICs used for *1 year* in MCS3 shows the effectiveness of the detection technique. In addition, PV2 is an extremely high variation compared to what is expected in practice (e.g., PV1).

With the same measurement temperature 25 °C, ±10 °C temperature variation is used in MCS4 during the aging process. The detection rate of ICs used for 1 month, 3 months, and 6 months in MCS4 are 90.6 %, 100 %, and 100 %, respectively. The fresh ICs' signature and the detected recycled ICs used for 3 months and 6 months are shown in Figure 14. Comparing Figure 14a with Figure 12c, it clear shows that the recycled ICs used for 3 months in MCS4 are closer to the signature than recycled ICs used for 3 months in MCS2. This phenomenon demonstrates that temperature variations could increase the path delay variations in fresh ICs and make it more difficult to detect recycled ICs. However, the 100 % detection rates of ICs used for 6 months in MCS4 demonstrates the effectiveness of our method with process and temperature variations.

Comparison Between SOA and PCA Figures 10 through 14 present some detailed results relating to using this technique on s38417 with SOA and PCA. Table 4, however, tabulates these results in addition to some other results obtained

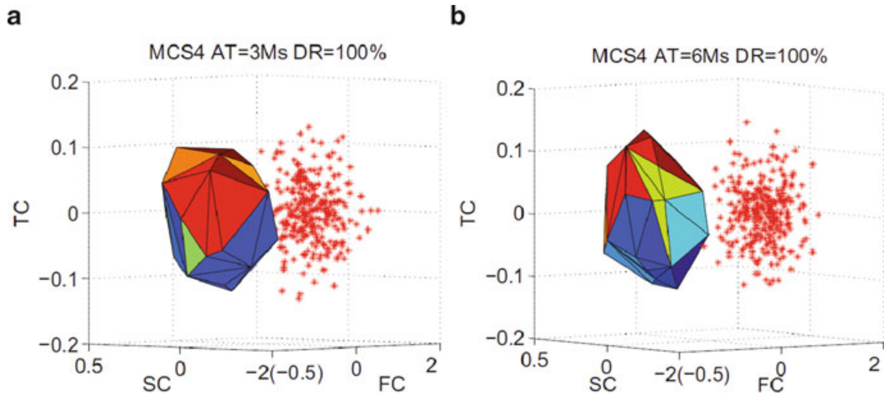


Fig. 14 PCA results of ICs with PV1 and ±10 °C temperature variations in MCS4 used for (**a**) 3 months and (**b**) 6 months

Table 4 Recycled IC detection rates for s38417

	SOA				PCA			
	1M	3M	6M	1Y	1M	3M	6M	1Y
MCS1	100 %	100 %	100 %	100 %	100 %	100 %	100 %	100 %
MCS2	78 %	96.7 %	99.7 %	100 %	96.3 %	100 %	100 %	100 %
MCS3	50.7 %	76.3 %	85.3 %	95.6 %	72.7 %	89.3 %	99.3 %	100 %
MCS4	67.7 %	93.3 %	98 %	100 %	90.6 %	100 %	100 %	100 %

using both statistical analysis approaches. These results clearly show that PCA is more effective than SOA when it comes to identifying ICs used for shorter periods of time.

Benchmark Analysis In addition to s38417, the ISCAS'89 benchmarks s9234 and s13027 were also simulated to demonstrate the efficiency of this technique on different designs. The process variation and temperature variation rates used in MCS4 were applied to these two benchmarks. The aging stress causing NBTI and HCI degradation in these benchmarks comes from random workloads. 300 MCS were run for each benchmark for a maximum 2 years of aging. The path selection method was also applied to these benchmarks, and 100 paths from each benchmark were used to run statistical data analysis using PCA.

Table 5 shows the recycled IC detection rate for all three benchmarks under MCS4 for up to a year of aging. The detection rate for ICs used for 3 months in the benchmarks s9234 and s13207 is 100 %, which matches the results obtained from s38417.

The results shown in this section clearly demonstrate that the recycled IC detection method using CTS with PCA is very effective, even in designs with large process and temperature variations.

Table 5 Recycled IC detection rates—benchmark comparison under MCS4 using PCA

Benckmark	1M	3M	6M	1Y
s9234	88 %	100 %	100 %	100 %
s13207	89.6 %	100 %	100 %	100 %
s38417	90.6 %	100 %	100 %	100 %

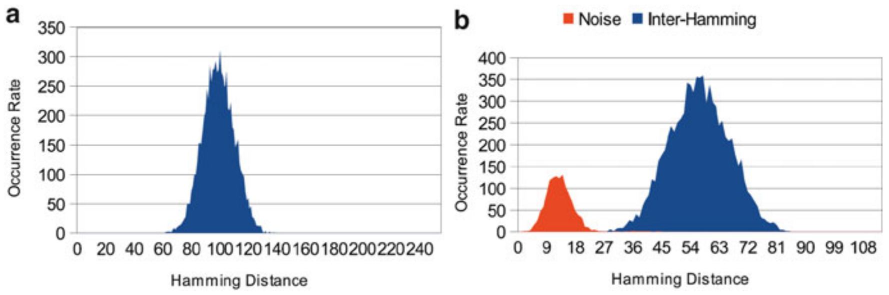


Fig. 15 Hamming distance analysis on 128 simulated s38417 circuits. (**a**) Unoptimized IDs (**b**) Optimized IDs with noise

6.2 Analysis of ID Generation for Detection of Remarked, Overproduced, and Cloned ICs

6.2.1 Simulation

To evaluate this methodology, a series of simulations in HSPICE were performed on an implementation of the ISCAS'89 benchmark s38417. This circuit was simulated at the 90 nm technology node. The CTS of the 256 top critical paths from the circuit were obtained and 128 Monte Carlo simulations were performed to add process variations. This provides 128 simulated ICs worth of data, with 256 data points for each IC. The PV1 in Table 2 is employed in the Monte Carlo simulations. All paths for all ICs were simulated over a range of temperatures from 23 °C to 27 °C to represent small deviations from room temperature.

Using the ID generation methodology described in Figure 9, 128 255-bit IDs were created from the simulation data. Figure 15a shows the results of Hamming distance analysis on these unoptimized IDs. The 255-bit IDs have, on average, a 99-bit or 39 % inter-Hamming difference. The fact that this average is less than 50 % means that some bit-positions in the IDs have a bias towards zero or one. If the optimization procedure described in Sect. 5.1.3 is applied, the average Hamming distance of the ID set improves and the size of the IDs is reduced, as shown in Figure 15b. The optimization process reduced the size of the IDs from 255 bits to 114 bits, and increased the average inter-Hamming distance from 39 to 50 %.

Figure 15b also shows the noise rates of the simulated IDs. Each IC was simulated at temperatures from 23 °C to 27 °C at 1 °C increments to represent small variations around room temperature. Hamming distance analysis was performed on each set of IDs coming from the same IC to see how many bits of the ID would change over the ID range. The average number of bits changing as a result of temperature was 13 bits, or about 11 %.

The Hamming distance distribution and noise rate distribution are both roughly normal distributions. By treating them both as normal distributions with their own averages and standard deviations, each set can be presented with a probability density function (PDF). These two PDFs can be used to find the Hamming distance at which it is equally likely that the two IDs are from different ICs or from the same IC. In the sets shown in Figure 15b, this Hamming distance is 27 bits. If two IDs are compared and have less than 27 different bits between them, they are most likely from the same IC. If two IDs are compared and have more than 27 different bits between them, they are most likely from different ICs.

6.2.2 FPGA Implementation

This methodology was also evaluated on 4490 nm Xilinx Spartan-3E FPGAs. The ISCAS'89 benchmark s9234 was implemented on our FPGAs, along with RAM for TDF pattern storage and structures for clock control. The IDs were sent from the FPGA boards to a computer for analysis by using an additional microcontroller. The paths in the FPGA were swept at the frequencies f_{1-16}, with $f_1 = \frac{1}{6.4ns}$ and $f_{16} = \frac{1}{3.4ns}$. The frequency step size was 200 ps. To obtain noise information, four of the implementations were measured eight times each. After selecting stable paths, the unoptimized IDs were 145-bit strings, with a Hamming distance distribution as shown in Figure 16a. The average inter-Hamming distance was 30 bits, or about 21 %. This is lower than in the simulation example. The optimization process reduces the size of the ID from 145 to 33 bits long, and the distribution of the inter-Hamming distances between these 33-bit IDs are shown in Figure 16b. By optimizing the IDs, the average inter-Hamming distance is changed from 30 out of 145 bits to 15 out of 33 bits, or about 45 %. In practice, a 33-bit ID would be short, but creating IDs of a greater size only requires measuring more paths.

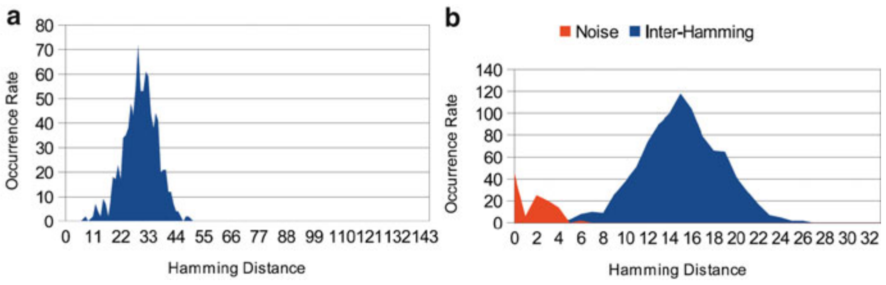


Fig. 16 Hamming distance analysis on 44 FPGA Implementations of s9234 circuit. (**a**) Unoptimized IDs (**b**) Optimized IDs with noise

Figure 16b also shows the noise rates for these IDs. Four different implementations of s9234 were measured 8 times each and the IDs from each implementation were compared to each other. On average, there was a 2 out of 33 bit difference between IDs from the same implementation, or about a 6 % difference.

Again, further analysis compares the noise and inter-Hamming rates. At a difference of 5.75 bits, it is equally likely that any two IDs being compared are from the same IC or from different ICs. IDs with a difference of less than 5.75 bits are probably from the same IC and IDs with a difference of more than 5.75 bits are probably from different ICs.

7 Summary

The counterfeiting of integrated circuits has become a major issue for the electronics industry. Currently, many researchers are endeavoring to develop effective and efficient methods to detect or prevent counterfeit ICs. In this chapter, two effective counterfeit detection methods using path-delay information are presented. In the first methods, a circuit timing signature (CTS) is generated for each IC by using the clock sweeping technique with conventional delay test patterns. The CTS from recycled ICs will be different from those from fresh ICs due to aging effects. The simulation results of different benchmarks with different process and temperature variations demonstrated the effectiveness of the detection method. In the second method, the CTS of each IC is converted to a unique ID (avoiding collisions) for a set of different implementations of the same circuit. The simulation and implementation results demonstrate that the average Hamming distance between the IDs is nearly 50 %, and the levels of noise are sufficiently low so that test engineers can distinguish between IDs from the same IC and IDs from different ICs. The generated IDs can be used to detect other counterfeit types, such as cloned, overproduced, and remarked ICs. With no additional hardware circuitry required, the clock sweeping technique introduces no additional overhead on area and power consumption.

References

1. Trust-HUB.: (2010). http://trust-hub.org/home
2. Cassell, J.: Reports of counterfeit parts quadruple since 2009. Challenging US Defence Industry and National Security
3. U.S. Senate Committee on Armed Services.: Inquiry into counterfeit electronic parts in the department of defence supply chain (2012)
4. U.S. Senate Committee on Armed Services.: Suspect counterfeit electronic parts can be found on internet purchasing platforms (2012)
5. US Congress.: National Defense Authorization Act for Fiscal Year (2012)

6. Guin, U., DiMase, D., Tehranipoor, M.: A comprehensive framework for counterfeit defect coverage analysis and detection assessment. J. Electron. Test. Theory Appl. (JETTA) **30**(1), 25–40 (2014)
7. US, Department of Commerce.: Defense industrial base assessment: counterfeit electronics (2010)
8. Guin, U., Forte, D., Tehranipoor, M.: Anti-counterfeit techniques: from design to resign. In: IEEE Microprocessor Test Verification (MTV) (2013)
9. Tehranipoor, M., Koushanfar, F.: A survey of hardware Trojan taxonomy and detection. IEEE Des. Test **27**(1), 10–25 (2010)
10. Brown, S.: Lifecycle of consumer digital products (2007)
11. Bureau of Industry and Security, U.S. Department of Commerce.: Defense industrial base assessment: counterfeit electronics (Jan 2010)
12. Tehranipoor, M., Wang, C.: Introduction to Hardware Security and Trust. Springer, Berlin/Heidelberg (2011)
13. Zhang, X., Tuzzio, N., Tehranipoor, M.: Identification of recovered ICs using fingerprints from a light-weight on-chip sensor. In: Proceedings of Design Automation Conference (DAC) (2012)
14. Guin, U., Forte, D., Tehranipoor, M.: Low-cost on-chip structures for combating die and IC recycling. In: Design Automation Conference (DAC) (2014)
15. Huang, K., Carulli, J., Makris, Y.: Parametric counterfeit IC detection via support vector machines. In: Proceedings of International Symposium on Fault and Defect Tolerance in VLSI Systems (DFT), pp. 7–12 (2012)
16. Zhang, X., Xiao, K., Tehranipoor, M.: Path-delay fingerprinting for identification of recovered ICs. In: Proceedings of International Symposium on Fault and Defect Tolerance in VLSI Systems (DFT) (Oct 2012)
17. Lofstrom, K., Daasch, W.R., Taylor, D.: IC identification circuit using device mismatch. In: Proceedings ISSCC 2000 (Feb 2000)
18. Pappu, R.: Physical one-way functions. Ph.D. Thesis, Massachusets Instutute of Tecnhology (2001)
19. Ozturk, E., Hammouri, G., Sunar, B.: Physical unclonable function with tristate buffers. In: Proceedings of ISCAS08, pp. 3194–3197 (2008)
20. Koushanfar, F., Qu, G., Potkonjak, M.: Intellectual property metering. In: Proceedings of 4th International Workshop Information Hiding, pp. 81–95 (2001)
21. Alkabani, Y., Koushanfar, F.: Active hardware metering for intellectual property protection and security. In: Proceedings of 16th USENIX Security Symposium, Usenix Association, pp. 291–306 (2007)
22. Contreras, G., Rahman, T., Tehranipoor, M.: Secure split-test for preventing IC piracy by untrusted foundry and assembly. In: Proceedings of International Symposium on Fault and Defect Tolerance in VLSI Systems (DFT) (2013)
23. Tuzzio, N., Xiao, K., Zhang, X., Tehranipoor, M.: A zero-overhead IC identification technique using clock sweeping and path delay analysis. In: IEEE GLSVLSI (2012)
24. Kimizuka, N., Yamamoto, T., Mogami, T., Yamaguchi, K., Imai, K., Horiuchi, T.: The impact of bias temperature instability for direct-tunneling ultra-thin gate oxide on mosfet scaling. In: VLSI Technology (1999)
25. Vattikonda, R., Wang, W., Cao, Y.: Modeling and minimization of pmos nbti effect for robust nanometer design. In: Proceedings of the 43rd Annual Conference on Design Automation (DAC'06), pp. 1047–1052 (2006)
26. Jiang, W., Le, H., Chung, J., Kopley, T., Marcoux, P., Dai, C.: Assessing circuit-level hot-carrier reliability. In: IEEE International Reliability Physics Symposium Proceedings, pp. 173–179 (1998)
27. Wu, L., et al.: Glacier: a hot carrier gate level circuit characterization and simulation system for vlsi design. In: Proceedings of IEEE International Symposium on Quality Electronic Design (ISQED), pp. 73–79 (2000)
28. Synopsys.: HSPICE user guide (2010)

29. Rabaey, J., Chandrakasan, A., Nikolic, B.: Digital Integrated Circuits: A Design Perspective, 2nd edn. Prentice Hall, Upper Saddle River (2003)
30. Bushnell, M., Agrawal, V.: Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits. Springer, Berlin/Heidelberg (2000)
31. Datta, R., Sebastine, A., Raghunathan, A., Abraham, J.: On-chip delay measurement for silicon debug. In: Proceedings of GLSVLSI' 04, pp. 145–148 (Apr 2004)
32. Ghosh, S., Bhunia, S., Raychowdhury, A., Roy, K.: A novel delay fault testing methodology using low-overhead built-in delay sensor. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **25**(12), 2934–2943 (2006)
33. Ghosh, A., Rao, R., Chuang, C., Brown, R.: On-chip process variation detection and compensation using delay and slew-rate monitoring circuits. In: Proceedings of ISQED'08, pp. 815–820 (Mar 2008)
34. Su, C., Chen, Y., Huang, M., Chen, G., Lee, C.: All digital built-in delay and crosstalk measurement for on-chip buses. In: Proceedings of DATE'00, pp. 527–531 (Mar 2004)
35. Wang, X., Tehranipoor, M., Datta, R.: Path-RO: A novel on-chip critical path delay measurement under process variations. In: Proceedings of International Conference on Computer-Aided Design (ICCAD) (Nov 2008)
36. INOVYS.: Test System for Complex SOCs. http://www.etesters.com/listing/40e8f648-a2d6-23b8-949b-4b3c005c86fb/OcelotZFP
37. Wang, S., Chen, J., Tehranipoor, M.: Representative critical reliability paths for low-cost and accurate on-chip aging evaluation. In: International Conference on Computer-Aided Design (ICCAD) (2012)
38. Chen, J., Wang, S., Tehranipoor, M.: Critical-reliability path identification and delay analysis. ACM J. Emerg. Technol. Comput. Syst. (JETC) **10**(2) (2014)
39. Jolliffe, I.T.: Principal Component Analysis, 2nd edn, pp. 150–165. Springer, Berlin/Heidelberg (2002)
40. Jin, Y., Makris, Y.: Hardware Trojan detection using path delay fingerprint. In: Proceedings of IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) (2008)
41. Xiao, K., Zhang, X., Tehranipoor, M.: A clock sweeping technique for detecting hardware Trojans impacting circuits delay. IEEE Des. Test **30**(2), 26–34 (2013)
42. Datta, R., Carpenter, G., Nowka, K., Abraham, J.: A scheme for on-chip timing characterization. In: Proceedings of VTS'06, pp. 24–29 (Apr 2004)

Hardware Trojan Detection in Analog/RF Integrated Circuits

Yier Jin, Dzmitry Maliuk, and Yiorgos Makris

Abstract Globalization of semiconductor manufacturing has brought about increasing concerns regarding possible infiltration of the Integrated Circuit (IC) supply chain by skilled and resourceful adversaries, with the intention of introducing malicious modifications (a.k.a hardware Trojans) which can be exploited to cause incorrect results, steal sensitive data, or even incapacitate a chip. While numerous prevention and detection solutions have been introduced in the recent, past, the vast majority of these efforts target digital circuits. Analog/RF ICs, however, are equally vulnerable and potentially even more attractive as attack targets, due to their wireless communication capabilities. Accordingly, in this chapter, we review existing research efforts in hardware Trojan detection in Analog/RF ICs. Specifically, using a wireless cryptographic IC as an experimentation platform, we demonstrate the effectiveness of side-channel fingerprinting along with advanced statistical analysis and machine learning methods in detecting hardware Trojans both after its manufacturing and after its deployment in its field of operation.

1 Introduction

The problem of maliciously intended modifications (a.k.a. hardware Trojans) in manufactured integrated circuits (ICs) has recently become of interest not only to academic researchers but also to governmental agencies and industrial entities [4]. Partly because of design outsourcing and migration of fabrication foundries to low-cost areas across the globe, and partly because of increased reliance on external hardware intellectual property (IP) and Electronic Design Automation (EDA)

Y. Jin
University of Central Florida, Orlando, FL 32816, USA
e-mail: yier.jin@eecs.ucf.edu

D. Maliuk
Yale University, New Haven, CT 06520, USA
e-mail: dzmitry.maliuk@gmail.com

Y. Makris (✉)
University of Texas at Dallas, Richardson, TX 75080, USA
e-mail: yiorgos.makris@utdallas.edu

software from various vendors, the integrated circuit supply chain is now considered far more vulnerable to such malicious modifications than ever before. Fears that skillful and resourceful adversaries may be able to compromise some stage of IC design and/or fabrication and insert Trojan hardware are becoming increasingly intense, as rumors about actual occurrence of such cases surface [4]. In essence, the fundamental concern is that hardware Trojan-infested chips may be capable of additional functionality which is unknown to the designer/vendor/customer and which can be exploited by the perpetrator after chip deployment. Evidently, depending on the field of application, the consequences of such attacks may range from minor inconvenience to major catastrophic events, especially since the intended target of such dubious ICs will most likely be a sensitive domain, such as financial, military, or other vital infrastructure.

While the severity of the potential implications of such a threat has fueled several research efforts towards better understanding and dealing with hardware Trojans both at the pre-silicon [14–16, 18, 22, 24, 27, 39] and at the post-silicon [6–10, 12, 20, 25, 26, 31–36, 40] stage, the vast majority of these efforts target traditional digital circuits. In contrast, this chapter will focus on the problem of hardware Trojans in the analog/RF domain and will also introduce hardware Trojan detection methods for wireless ICs. Similar to digital circuits, analog/RF ICs are now prevalent in electronic systems, facilitating industrial control and wireless communication and becoming an inseparable part of modern everyday activities. At the same time, analog/RF ICs (and, by extension, the integrated systems containing analog/RF modules) are particularly vulnerable and constitute a very appealing target for hardware Trojan attacks; indeed, since such circuits receive and transmit information over public wireless channels, the attacker does not need to obtain physical access to their input/output space, making such attacks far more realistic. Moreover, most modern communication systems employ some form of encryption in order to protect the privacy of the information that is communicated over the public channel. Interestingly, while this provides the user with an—often misleading—sense of security, it also entices attackers, who know that valuable secret information (e.g. the encryption key) is stored on these devices. Therefore, development of pertinent Trojan hardware mitigation methods for analog/RF ICs is equally (if not more) critical as with their digital counterparts.

Toward this end, this chapter studies the threat of hardware Trojans specifically within the context of analog/RF ICs and examines remedies to ensure their trustworthiness both during the manufacturing testing process and after their deployment in their field of operation. Through the material introduced in this chapter we seek to achieve the following objectives:

- Delineate the threat and potential impact of hardware Trojans in analog/RF ICs. Specifically, we will focus on vulnerability introduced by the margins that are typically allowed in the transmission parameters in order to deal with process variations and we will show that these margins can be exploited in order to gain control of a chip and/or leak sensitive information. The trade-off between the level of harm that these hardware Trojans may incur and the impact on area/power/performance, which is strongly correlated to their detection susceptibility, will also be investigated.

- Elucidate the shortcomings of existing test methods in exposing hardware Trojans in the analog/RF IC domains. Since analog/RF Trojans do not change the functionality of the chip, they are very difficult to be detected by traditional manufacturing testing methods. The effectiveness of existing hardware Trojan detection methods introduced in the digital domain will also be investigated.
- Devise efficient hardware Trojan detection methods based on statistical analysis and machine learning, specifically for analog/RF ICs. The effect of a carefully designed hardware Trojan is expected to be hidden within the parametric design margins, making side channel information of a Trojan-infested chip appear perfectly legitimate if examined in isolation. However, for the hardware Trojan to be of utility to the attacker, it needs to impose some form of structure in the transmission signals and/or other side-channel signals, through which remote commands will be issued or secret information will be leaked. Statistical analysis methods can therefore be used to detect the existence of this added structure and machine learning (i.e. trained classifiers) can be used to distinguish between Trojan-free and Trojan-infested chip populations.

2 Hardware Trojans in Wireless ICs

Using as an experimentation vehicle a simple wireless cryptographic circuit and two example hardware Trojans which were specifically designed to attack wireless ICs [21], we will demonstrate the following three key findings:

- **Attack Complexity:** Minor modifications to a wireless cryptographic chip suffice to leak secret information. The vulnerability of such chips stems partly from the fact that they transmit over a public wireless channel. Their true Achilles heel, however, is the fundamentally analog nature of a wireless transmission, which entails several continuous parameters (e.g. amplitude, frequency, phase, etc.). In order to tolerate variations due to fabrication process and/or operating conditions, specifications for these parameters are defined as windows of acceptable performances rather than exact values. As a result, a hardware Trojan can hide additional information within the tolerance margins of such continuous entities and secretly transmit it. While such transmissions abide by all specifications and appear to be perfectly legitimate, an adversary who knows the structure of the additional information will be able to extract it.
- **Detection Difficulty:** Evading detection by traditional manufacturing test methods is trivial. The functionality of the digital part of the chip in normal operation mode and in test mode can be preserved despite the addition of the hardware Trojan; hence, no structural (i.e. scan-based) or functional tests (or even enhanced functional tests for hardware Trojan-detection) will fail in a fault-free but Trojan-infested chip. Similarly, since the analog functionality of the chip is left intact,

all analog/RF specification tests will pass. Furthermore, since the leaked information is hidden within the allowed transmission specification margins, system-level functional tests will also pass. Existing side-channel fingerprint generation and checking methods, at least in their original form, also fall short in detecting hardware Trojans in wireless cryptographic ICs.

- **Possible Solution:** Despite the fact that hardware Trojans can be hidden within the process variation margins of a wireless cryptographic chip and may not be exposed through any of the above methods, it may still be possible to detect them. Effective hardware Trojans must impose a specific structure on the transmission parameters, which the attacker leverages to snoop the secret key. While this structure is not known to the defender, advanced statistical analysis of these parameters may be sufficient to reveal its existence and, thereby, expose the hardware Trojan. Since the attacker does not know what data will be collected or how it will be analyzed, this method is difficult to evade. In other words, the element of surprise by the attacker, who picks the structure of the hidden data, is counteracted by a similar element of surprise by the defender, who picks the measurements and the statistical analysis method.

3 Pre-Deployment Hardware Trojan Detection

The most common threat model adopted in hardware Trojan research assumes that the culprit is either at the foundry or at design houses where third party intellectual property (IP) is acquired from. In either case, once silicon is obtained and before it is shipped to customers, it is essential to test not only for manufacturing defects (which is the objective of VLSI testing) but also for hardware Trojans. Therefore, we first discuss the problem of pre-deployment hardware Trojan detection in analog/RF ICs, wherein we can exercise the device under test in a controlled environment with pre-specified stimuli.

3.1 Experimentation Vehicle

The experimentation vehicle used to elucidate the problem of hardware Trojans in analog/RF ICs is shown in Fig. 1. This is a mixed-signal wireless cryptographic IC, capable of encrypting and broadcasting data, which can be used in secure data transmission over open channels. The digital part includes a pipelined Digital Encryption Standard (DES) core [2], an output buffer and a serializer, which serves as the interface between the digital and the analog part. The analog part is an Ultra-Wide-Band (UWB) transmitter.

The DES core in the chip is a performance-optimized design with 16 encryption blocks in a pipeline structure. Each block can independently run the Feistel function f, which is the central part of the DES algorithm. A fully pipelined key

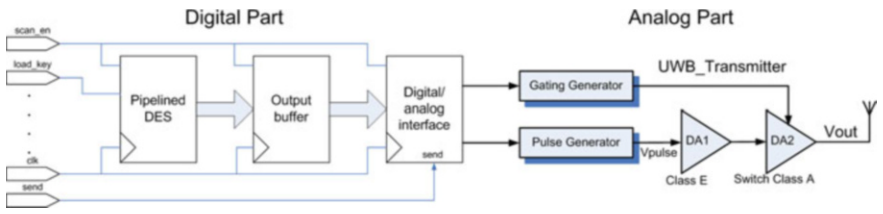


Fig. 1 Block diagram of example wireless cryptographic integrated circuit

generation module is designed to operate in parallel with these encryption blocks. In order to achieve high operating frequency, the initial permutation and inverse initial permutation of the plaintext are handled through hard-wiring, with no logic circuitry involved. The widths of the input and output data are both 64 bits, which is the length of a plaintext/ciphertext block. The output buffer is a First-In First-Out (FIFO) structure of 64-bit words, which supports reading and writing speeds commensurate with the performance of the pipelined DES core. The digital/analog interface converts the 64-bit data block from the buffer into a serial bit stream and passes it on to the UWB transmitter. The interface also adjusts the data-sending frequency to ensure signal integrity in this mixed-signal design. A pulse on the `send` primary input passes the contents of the output buffer to the interface and finally to the UWB transmitter for broadcasting. The UWB transmitter [41] consists of a pulse signal generator, a gating signal generator and two driver amplifiers (DAs) and can transmit data over a wide spectrum of frequency bands with very low power consumption and high data rate. The UWB transmitter is in active mode and transmits a high frequency signal when the information bit to be transmitted is ‘1’, otherwise it is in idle mode.

The chip is designed in TSMC CL013G .13μm CMOS technology process [1]. The digital part runs at a frequency of 75MHz and the UWB transmitter has a data rate that exceeds 50 Mbps. Tests for the digital part cover both stuck-at and delay faults using a full-scan chain of Enhanced Scan Flip-Flops [17]. For the analog part, besides the traditional specification tests, the spectrum of the output pulse sequence of the DA chain at a data transmission rate of 50 Mbps is also measured [41]. System-level functional tests involve randomly generated patterns which are encrypted and broadcasted by the UWB transmitter. A receiver decrypts the ciphertext and compares to the expected plaintext, in order to detect any discrepancies.

Figure 2 shows a simulation example of a typical transmission of a 64-bit block of ciphertext and a magnified view of the transmission signal when a ‘1’ bit is broadcasted. UWB specification calls for a transmission frequency between 3.1 GHz and 10.6 GHz. The specifications for this particular implementation define its frequency between 4 GHz and 6 GHz. Transmitting a ‘1’ bit involves between 5 and 7 peaks of amplitude over 300uW with at least one of them over 900uW. The actual performances of each individual chip will vary, depending on the fabrication process variations. For example, the response of the circuit instance shown in

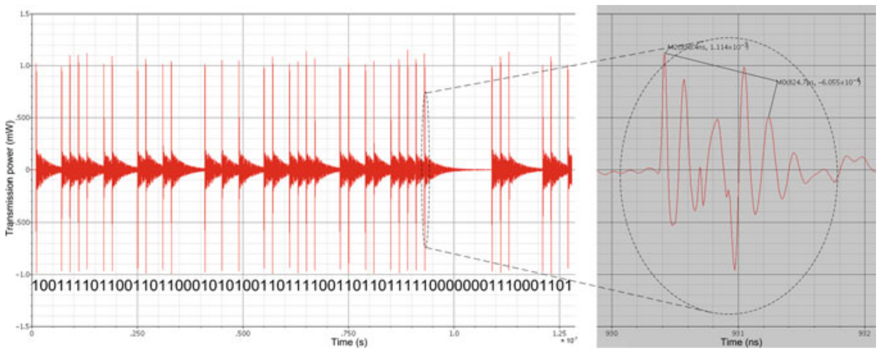


Fig. 2 Example of 64-bit ciphertext block transmission

the figure, which was randomly picked from a population of 200 chips generated through a Monte Carlo Spice simulation with 5 % process variation on all transistor parameters, operates at a frequency of 4.8 GHz and involves 5 peaks of amplitude over 300uW with the largest measuring at 1114uW.

3.2 Hardware Trojans

Two hardware Trojans are designed which, through minute modifications, are capable of leaking the encryption key by hiding it in the wireless transmission parameter (i.e. amplitude or frequency) margins allowed in the design specifications in order to deal with process variations. Thus, they ensure that the circuit continues to comply to all of its functional specifications. The working principle of these Trojans is simple: extract one bit at a time from the 56-bit encryption key, which is stored in the DES core, and leak it by hiding it in one 64-bit block of transmitted data. After 56 ciphertext blocks are transmitted, the entire key will have been broadcasted.

Implementation Details Each hardware Trojan involves two modifications. The first modification, which is shown in Fig. 3a, is common to both hardware Trojans and aims to extract the encryption key from the DES core. The second modification, which is shown in Fig. 3b, is different for each of the two hardware Trojan and aims to manipulate the transmission amplitude or frequency in order to leak the key through the wireless channel.

The key extraction modification exploits the ability of Enhanced Scan Flip-Flops to store two bits, one in the D flip-flop and one in the follow-up latch, so that back-to-back vectors can be applied for the purpose of detecting delay faults when the circuit is in test mode [17]. During normal operation, however, the latches are transparent, essentially holding the same information as the D flip-flops. In the example circuit, the 56-bit encryption key is stored in a sequence of 56 Enhanced

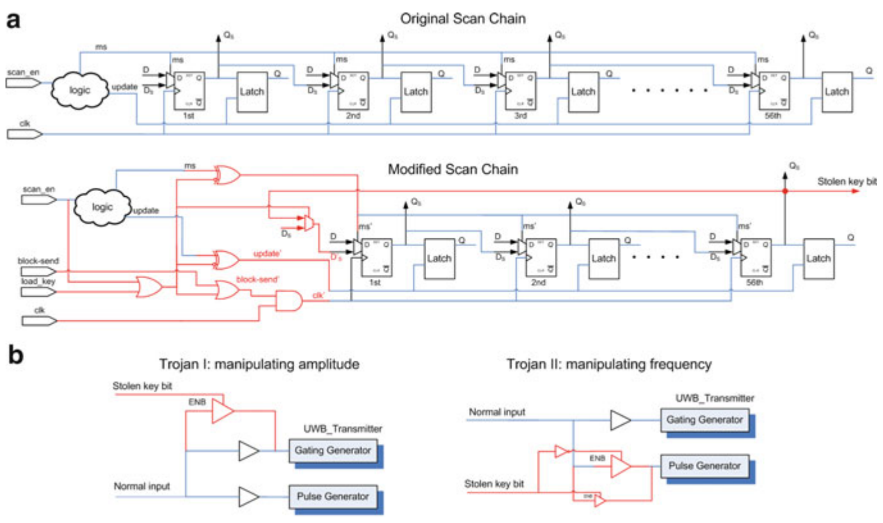


Fig. 3 (**a**) Extracting the key bitwise, through a rotator made out of the 56 enhanced scan flip-flops where it is stored. (**b**) Broadcasting the stolen key bit by manipulating the amplitude or the frequency of the UWB transmission

Scan Flip Flops which are serially connected in a scan chain, as shown in the top part of Fig. 3a. The basic idea for extracting the secret key is to store it only in the latches of the Enhanced Scan Flip Flops and reuse the D flip-flops to create a 56-bit rotator. Initially, when the key is loaded by the user, both the flip-flops and the latches hold the correct bits. Then, every time a data block is transmitted, the last bit of this rotator is extracted and hidden in the transmission, while the rotator shifts its contents by one position. Only the D flip-flops of the Enhanced Scan Flip Flops hold a rotated version of the key, while the follow-up latches continue to hold the correct version, so that the ciphertext is correctly produced. Simple control logic consisting of a few gates, shown in red color in the bottom part of Fig. 3a, suffices for this purpose.

The key transmission modification receives the stolen bit and based on its value modifies the transmission signal in one of two ways. The first option (Type-I), shown on the left side of Fig. 3b, manipulates the transmission amplitude; when the stolen key bit is ‘1’, an additional driver strengthens the legitimate transmission signal before it reaches the gating generator, thereby slightly increasing the transmission amplitude. Figure 4a shows the corresponding impact on the signal transmitted by the example circuit instance used in Fig. 2. In this case, the amplitude increases from 1114uW to 1235uW, but the frequency remains at 4.8 GHz. The second option (Type-II), shown on the right side of Fig˙ 3b, manipulates the transmission frequency; when the stolen key bit is ‘1’, the original buffer is bypassed and an alternative buffer is used to delay the output of the pulse generator, thereby slightly increasing the transmission frequency. Figure 4b shows the corresponding impact on

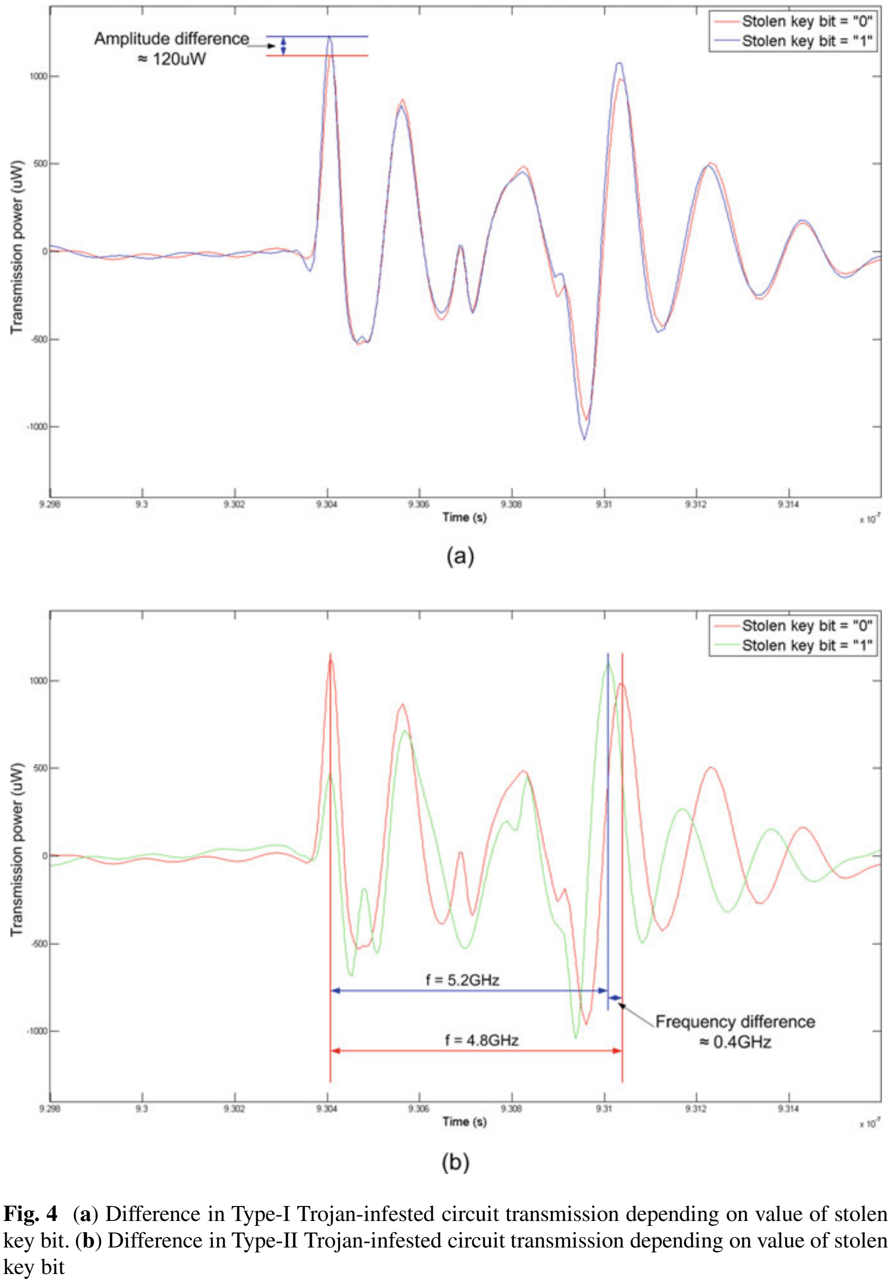


Fig. 4 (**a**) Difference in Type-I Trojan-infested circuit transmission depending on value of stolen key bit. (**b**) Difference in Type-II Trojan-infested circuit transmission depending on value of stolen key bit

the signal transmitted by the example circuit instance used in Fig. 2. In this case, the frequency increases from 4.8 GHz to 5.2 GHz but the amplitude remains at 1105uW. In both cases, when the stolen key bit is ‘0’, no change occurs in the transmitted signal.

The overall area overhead incurred by each of the above Trojans is around 0.02 % of the digital part of the chip. This figure assumes that the storage elements holding the secret key are Enhanced Scan Flip Flops which are connected in sequence. If this is not the case and a separate 56-bit rotator needs to be added, the area overhead still remains well below 0.4 % of the digital part of the chip.

Secret Information Extraction Figures 4a,b show the transmission power waveform of a Type-I and a Type-II Trojan-infested chip, respectively, when the stolen key bit transmitted along with the legitimate signal is ‘1’, as well as when it is ‘0’. Evidently, in the Type-I Trojan-infested chip, the difference in the stolen key bit value is reflected as a difference of 120uW in the maximum amplitude. Similarly, in the Type-II Trojan-infested chip, the difference in the stolen key bit value is reflected as a 0.4 GHz difference in the frequency. Both of these differences are well within the margins allowed for process variations and operating condition fluctuations and would not raise any suspicion. While the attacker does not know a priori the exact amplitude or frequency levels in each of the two cases, the fact that this difference is always present suffices for extracting the secret key. All the attacker needs to do is listen to the wireless channel to observe these two different amplitude or frequency levels, which correspond to a stolen key bit of ‘1’ and a stolen key bit of ‘0’, respectively. Once these two levels are known, listening to 56 consecutive transmission blocks reveals a rotated version of the 56 bits of the encryption key. Using this information, the attacker needs at most 56 attempts (i.e. all rotations of the extracted 56 bits) to decrypt the transmitted ciphertext.

3.3 Evaluation of Existing Test and Trojan Detection Methods

The mechanism through which the two hardware Trojan examples leak the secret information over the wireless channel allows them to evade detection not only by traditional manufacturing testing but also from previously proposed Trojan detection methods.

Functional, Structural, and Enhanced Testing The hardware Trojan examples do not alter the functionality of the digital part of the circuit. In normal operation, the enhanced scan flip-flops that hold the key bits are loaded appropriately. Numerous randomly generated functional test vectors are simulated to verify the correctness of the produced ciphertext. In test mode, the scan chain also operates as expected. To demonstrate that structural tests do not detect these hardware Trojans, a standard industrial ATPG tool is used to generate test vectors for all stuck-at and delay faults in the Trojan-free circuit. These tests are simulated on the two Trojan-infested circuits. As expected, all tests passed. Enhancing the test set with further vectors that exercise rare events [35, 40] is also ineffective, since the hardware Trojans do not affect the digital functionality. The analog portion is not modified and, therefore, it also passes the traditional specification-based analog/RF test.

System-Level Testing System-level tests examining the parameters of the wireless transmission also fail to expose the hardware Trojans, since the structure added by the leaked information is hidden within the margins allowed for process variations. To demonstrate this, we measured the transmission power of 200 genuine (i.e. Trojan-free) chips, 100 chips infested with a Type-I hardware Trojan and 100 chips infested with a Type-II hardware Trojan, which we generated using Monte Carlo Spice-level simulation assuming 5 % process variations on all circuit parameters. Figure 5a plots the transmission power when a '1' is transmitted by half of these chips, as well as the $\mu \pm 3\sigma$ envelope of the transmission power when a '1' is transmitted by the other half of these chips. Figures 12b,c plot the transmission power when a '1' is transmitted by the Type-I and Type-II Trojan infested chips, respectively. Evidently, given any one of these transmission power plots, it is not possible to distinguish whether it comes from a Trojan-free or a Trojan-infested chip.

Local Current Traces An interesting hardware Trojan detection method based on local current traces is presented in [33, 34]. This test strategy detects anomalies introduced by the Trojan in the currents measured at the power ports and takes into account process and operating conditions variations. The authors demonstrate that their method can detect Trojans of size as small as 2 % of the power grid. In order to implement this method in the design, the chip needs to be divided into at least 20 power grids with at least 30 uniformly located power ports. The availability of these power ports is a serious obstacle to implementing this method. Furthermore, a capable attacker would probably observe the existence of these power ports and could possibly invent countermeasures to prevent the injected hardware Trojans from becoming visible through these ports.

Global Power Traces In [5], the authors use global power consumption traces to distinguish between Trojan-free and Trojan-infested chips. The method employs statistical analysis of the Eigenvalue spectrum and can effectively detect hardware Trojans occupying 0.12 % of the total circuit area, assuming process variation in the order of 5 %. But when the hardware Trojan area is reduced to only 0.01 % and the process variation is increased to 7.5 %, false alarms start to appear. Considering the very low area overhead of the hardware Trojans (i.e. 0.02 %) and based on the limitations outlined in [5], it is unlikely that statistical analysis of the total power consumption will expose them. Indeed, even when this method is applied to the power traces of the digital part only,[1] wherein the hardware Trojans are hidden, it was not possible to effectively distinguish between Trojan-free and

[1]Mixed-signal SoCs typically have separate power ports for the analog and the digital parts.

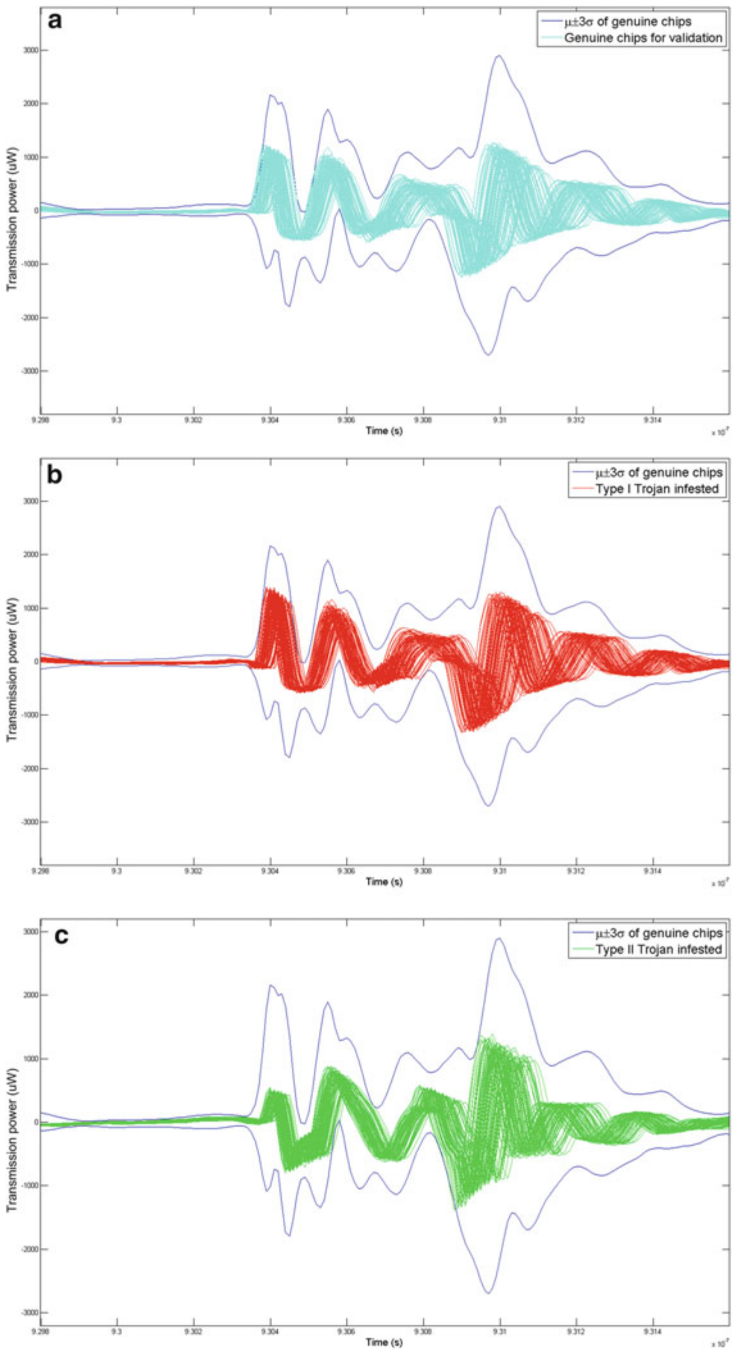


Fig. 5 (**a**) $\mu \pm 3\sigma$ transmission power envelope of 100 Trojan-free chips and transmission power of another 100 Trojan-free chips. (**b**) Transmission power of 100 Type-I Trojan-infested chips. (**c**) Transmission power of 100 Type-II Trojan-infested chips

Trojan-infested chips in any Eigenvalue sub-space. Nevertheless, as mentioned in [5], other parameters may still prove effective. In fact, the solution used in the following section employs a similar statistical analysis of the wireless transmission power.

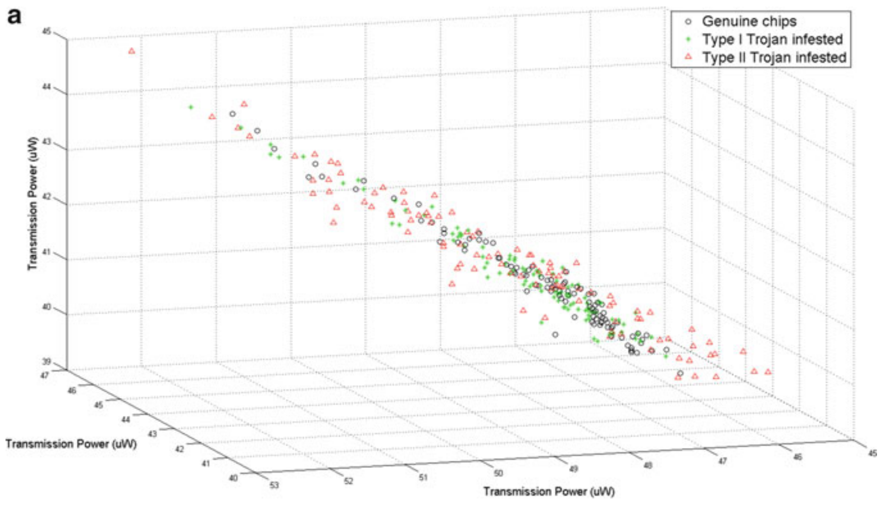
Path Delay Traces A similar statistical method proposed in [20] utilizes path delay fingerprints to differentiate Trojan-free from Trojan-infested chips. While the hardware Trojan examples add some delay to a small number of paths in the digital part of the circuit, the impact is too small to be observed. Even if those paths related to the encryption key are checked, the complexity of the pipelined encryption circuitry provides enough margin to hide the added delay. To verify this, the path delay based Trojan detection method was applied assuming process variations in the range of 5 % but it was unable to identify the existence of hardware Trojans.

3.4 Statistical Analysis to the Rescue

While the structure added to the transmitted signal for the attacker to extract the stolen key leaves individual transmissions within the acceptable specification boundaries, it enables the possibility that such hardware Trojans can be exposed through statistical analysis of the transmission parameters.

To demonstrate this principle, a measurement the total transmission power is used for broadcasting one block of data (i.e. 64-bits). For 100 Type-I Trojan-infected, 100 Type-II Trojan-infected, and half of the 200 Trojan-free circuit instances which are generated via Monte Carlo simulation with 5 % process variations, the total transmission power is measured when transmitting each of six randomly selected blocks (the same for all circuits). Of course, the Trojan-infested chips also leak one key bit during each of the six transmissions, half of which are set to ‘1’. All six measurements for all genuine and all Trojan-infested chips are within the acceptable specification range. Even when the three chip populations are projected on the six-dimensional space of these measurements, it is impossible to distinguish them since they fall upon each other. Figure 6a shows a projection of the three populations on three of these dimensions. Evidently, separating the genuine from the Trojan-infested populations in this space is not possible. The situation is similar for any other subset of three measurements.

However, running a Principal Component Analysis (PCA) on these measurements reveals that the structure of the genuine chip data is different than the structure of the Trojan-infested chip data. Figure 6b shows a projection of the three populations on the three principal components of the data, clearly revealing that they are separable in this space. Therefore, the trusted boundary is defined as a simple minimum volume enclosing ellipsoid (MVEE [30]) which encompasses the genuine population. Then, any chip whose footprint on the space of the selected three principal components does not fall within the trusted boundary will be discarded as suspicious. In the example, this method detects all Type-I and Type-II Trojan-infested chips without inadvertently discarding any genuine chips.



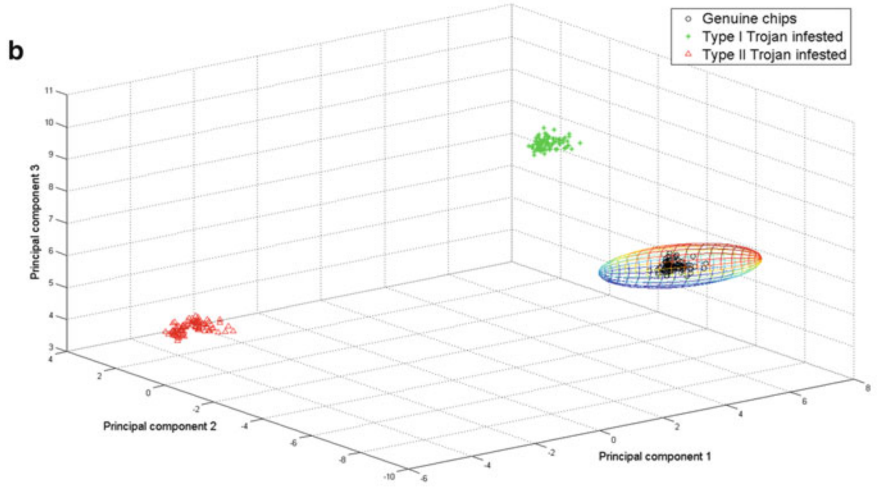


Fig. 6 (**a**) Projection of genuine and Trojan-infested chip populations on three out of six transmission power measurement. (**b**) Projection of genuine and Trojan-infested chip populations on three principal components of six transmission power measurements

Given the small number of transmission parameters (or combinations thereof) wherein the attacker can hide the added structure, as well as the large number of measurements that the defender can utilize to identify statistical discrepancies, the defender can easily detect the inserted hardware Trojan. Finally, similar statistical analysis and machine learning-based methods involving parametric measurements have been previously employed successfully for the purpose of manufacturing testing [38] and radiometric fingerprinting [11] of analog/RF circuits. However, this is the first attempt to apply such methods towards hardware Trojan detection in wireless cryptographic ICs or aanalog/RF ICs in general.

4 Post-Deployment Hardware Trojan Detection

While the aforementioned side-channel fingerprinting method can be very effective in detecting hardware Trojans prior to IC deployment, it relies on the assumption that the Trojan is active at test time. Hence, it fails to detect dormant hardware Trojans which are activated only *after* an IC is deployed in its field of operation, through a lapsed-time counter or an external trigger [19]. Therefore, continuing to evaluate trustworthiness after deployment through on-chip support for hardware Trojan detection is equally important. To this end, in this section we introduce a general post-deployment hardware Trojan detection architecture [23], which is based on-chip measurement acquisition and classification, and we demonstrate its effectiveness on the wireless cryptographic IC experimentation vehicle.

4.1 Proposed Trust Evaluation Architecture

The proposed architecture for post-deployment trust evaluation is shown in Fig. 7. The overall idea is fairly straightforward: after the circuit is deployed, the end-user can trigger the trust evaluation procedure at any time; during trust evaluation,

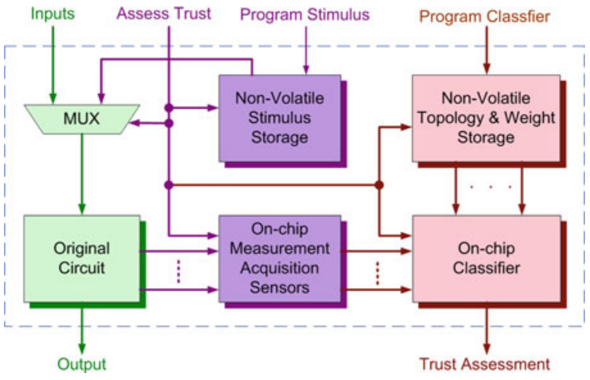


Fig. 7 Proposed post-deployment trust evaluation architecture

on-chip resources are used to apply a known stimulus to the circuit and to obtain parametric measurements, which are subsequently assessed on-chip to decide whether the circuit is operating within a trusted region. To this end, several components are added to the chip, along with the original circuit:

- A programmable on-chip non-volatile stimulus storage component (i.e., Flash, EEPROM, or OTPROM) and a multiplexer through which the known necessary excitation stimulus is provided to the circuit.
- Measurement acquisition sensors, to obtain the parametric signature of the circuit in response to the known stimulus.
- An on-chip classifier, to assess the parametric signature obtained via the sensors and to decide whether the circuit operation is trusted or not.
- Programmable on-chip non-volatile storage for programming the topology and the weights that define the region accepted as trusted by the classifier.

The programmability and non-volatility are required, so that the actual stimulus, the topology of the classifier, and the region accepted as trusted are stored on the chip only after it is fabricated. Thereby, a potential attacker is not privy to this information. While the attacker may be able to understand what parameters are being measured, without knowledge of the stimulus, the actual structure of the classifier and the definition of the trusted region, it will be very difficult to design a hardware Trojan that evades detection. In essence, the proposed architecture counteracts the element of surprise possessed by the attacker (i.e., the ability to choose the location, functionality, and time of activation of the hardware Trojan) by a similar element of surprise possessed by the defender (i.e., the ability to choose the type of parametric signature, the method and bounds for assessing its trustworthiness, and the time of trust evaluation).

4.2 Experimentation Vehicle

4.2.1 Target Circuit

The experimental platform which is used to demonstrate the effectiveness of the proposed post-deployment Trojan detection method is an extension of the mixed-signal wireless cryptographic IC [21] which was used in the previous section. We remind that this chip takes plain-text at its input, encrypts it using an on-chip stored key, and then transmits the cipher-text on a public wireless channel. Figure 8 shows the basic architecture of the entire platform, which is divided into three parts: (i) the digital part, which includes a pipelined Digital Encryption Standard (DES) core, an output buffer, and a serializer serving as the interface between the digital and analog parts, (ii) the analog part, which is an ultrawide-band (UWB) transmitter, and (iii) the on-chip resources, which are added for the purpose of post-deployment hardware Trojan detection. These include an on-chip non-volatile serial-in parallel-out 64-bit register to hold the trust evaluation stimulus,

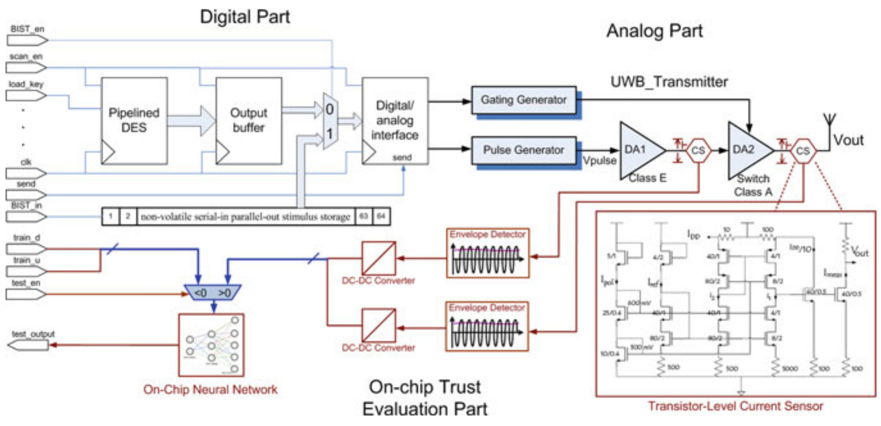


Fig. 8 Architecture of wireless cryptographic IC experimental platform

two current sensors along with envelop detectors and DC-DC converters to obtain the side-channel fingerprint of the chip, and a neural network to classify it as trusted or untrusted. The current experimentation platform consists of SPICE-level simulation models for all components, except for the neural classifier. The latter is emulated through a programmable analog neural network experimentation chip to demonstrate, in silicon, the ability to detect hardware Trojans.

4.2.2 On-Chip Trust Evaluation Resources

The on-chip trust evaluation part performs two tasks, namely parametric measurement acquisition and data classification. Parametric measurements are obtained via on-chip sensors in response to a known stimulus, which is also stored on-chip using a non-volatile serial-in parallel-out shift (SIPO) register, as shown in Fig. 8. The `BIST_in` signal is used to fill in the 64-bit wide register with a value *after* fabrication and prior to deployment. Another `BIST_en` signal controls the data flow to the digital/analog interface. When `BIST_en` is ‘0’, the input of the interface is the ciphertext to be sent by the UWB transmitter while when it is ‘1’, the pattern stored in the SIPO register is sent to the UWB transmitter, in order to perform trust evaluation.

In this platform, two current measurements obtained from the UWB transmitter are used for trust evaluation. In order to lower area overhead and increase accuracy/stability of the measured currents, a robust CMOS built-in current sensor (BICS) is implemented [13]. The transistor-level structure of this current sensor can be seen in the blow-out part of Fig. 8. The output of the BICS is a high frequency signal which is then converted to a DC voltage through a CMOS envelope detector [3]. Both the current sensor and the envelope detector are CMOS designs so that they are compatible with other parts of the circuit. A DC-DC converter is then used to match the measurement to the input range of the circuit that will perform data classification (i.e. the on-chip neural network).

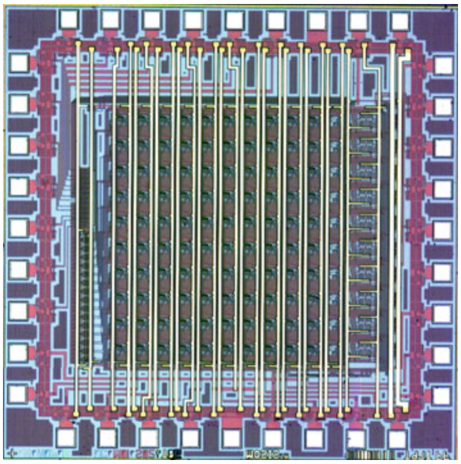


Fig. 9 Micrograph of analog neural network chip

4.2.3 On-Chip Classifier

To demonstrate in silicon that an on-chip classifier can, indeed, detect a hardware Trojan upon its activation in the operation field, an analog neural network experimentation chip is employed [28]. Using this programmable chip, artificial neural networks are implemented, which are then trained to learn (through a training set of chips) the mapping between the current measurements obtained from the two BICS integrated inside the UWB transmitter, and the trusted operation region. The trained neural networks can then be evaluated with respect to their capability to detect Trojan-infested chips using a validation set. Note that an analog VLSI implementation of the neural classifier is necessary in order to contain the area and power overhead of the proposed trust evaluation.

Figure 9 shows the stand-alone version of the programmable analog neural network chip which is used in the platform. This chip serves as a flexible platform for the experiments by virtue of two properties: *trainability*, which allows it to learn complex boundaries from the training set, and *reconfigurability*, which is used to adjust the number of hidden neurons to match the complexity of the target task. The possible topologies include all 2-layer networks within the available number of on-chip synapses and neurons. As will be shown later, network topologies with very small number of hidden neurons are sufficient to meet both the accuracy requirements to differentiate Trojan-infested chips from genuine chips and the low overhead requirements. Figure 10 illustrates the block-level schematic of the circuit implementation in the neural network chip. The circuit consists of a matrix of synaptic blocks (S) and neurons (N). The synapses represent mixed-signal devices, in the sense that they conduct all computations in analog form while their weights are implemented as digital words stored in a local memory. The results of synapse multiplication are summed and fed to the corresponding neuron, which performs a squashing function and produces an output either to the next layer or the primary output. The architecture is very modular and can easily be expanded to any number of neurons and inputs within the available silicon area [29].

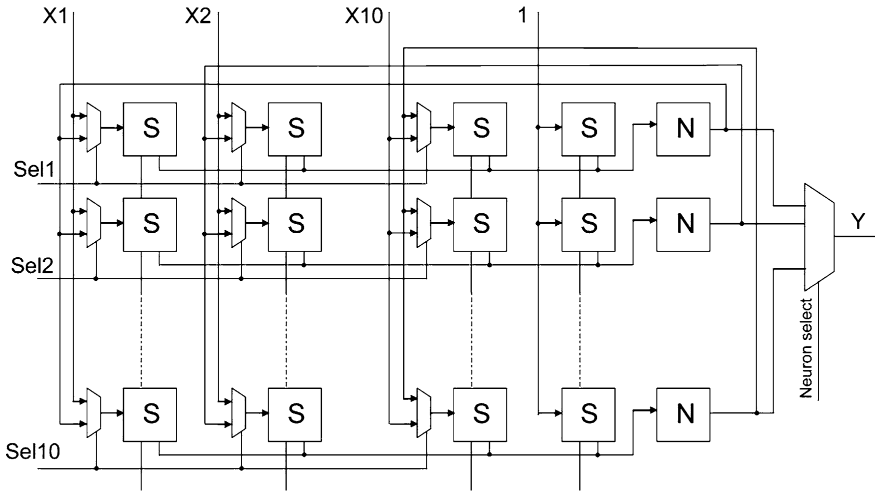


Fig. 10 Reconfigurable neural network architecture

4.2.4 Hardware Trojans

In addition to the Trojan-free circuit, two alternative hardware Trojan-infested variants of the wireless cryptographic IC are also designed. These are of similar structure and working principle to the Trojans introduced in the previous section, with the exception that both Trojans are dormant during the testing stage and are only activated after deployment.[2] As before, through simple modifications, when activated these hardware Trojans leak the encryption key by hiding it in the wireless transmission amplitude or frequency margins allowed due to process variations; thus, they ensure that the circuit continues to comply to all of its functional specifications and, thereby, evade testing both on the digital and on the analog side.

Figures 11a, b show the transmission power waveform of a Type-I and a Type-II Trojan-infested chip, respectively, when the Trojan is activated and the stolen bit is ‘1’, as well as when the Trojan is dormant (in which case, the stolen bit value is irrelevant). Evidently, in the Type-I Trojan-infested chip, the activation of the Trojan will alter the maximum amplitude by as much as 380uW from which attackers can differentiate a logic ‘1’ or logic ‘0’ value for the stolen key bit. Similarly, in the Type-II Trojan-infested chip, the difference in the stolen key bit value is reflected as a 0.4 GHz difference in the frequency when the Trojan is activated. Both of these differences are well within the margins allowed for process variations and operating condition fluctuations and would not raise any suspicion. While the attacker does not know a priori the exact amplitude or frequency levels in each of the two cases, the fact that this difference is always present suffices for extracting the secret key.

[2]Interested readers are referred to [19] for a relevant discussion on Trojan triggering.

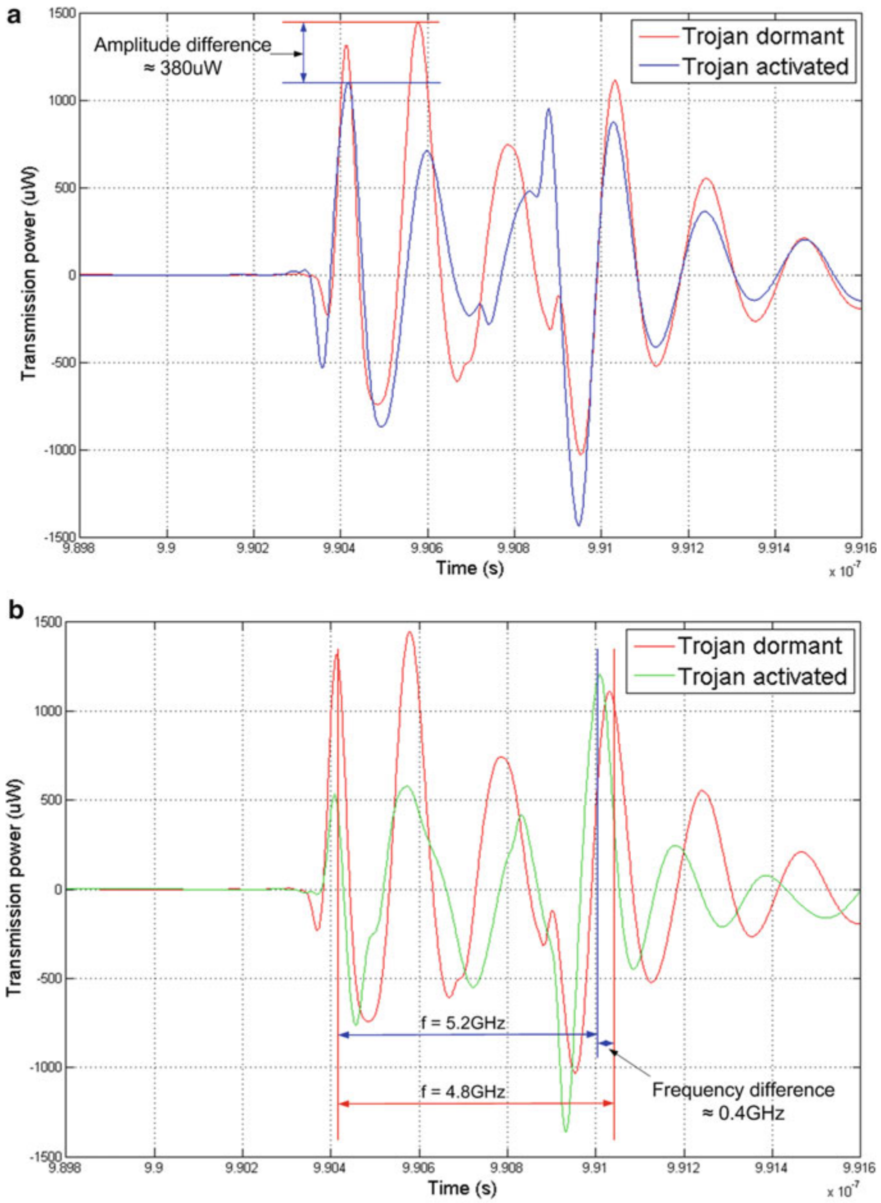


Fig. 11 (**a**) Difference in Type-I Trojan-infested circuit transmission when Trojan is dormant and activated. (**b**) Difference in Type-II Trojan-infested circuit transmission when Trojan is dormant and activated

All the attacker needs to do is listen to the wireless channel to observe these two different amplitude or frequency levels, which correspond to a stolen key bit of ‘1’ and a stolen key bit of ‘0’, respectively, after the Trojan is activated. Once these two levels are known, listening to 56 consecutive transmission blocks reveals a rotated version of the 56 bits of the encryption key. Using this information, the attacker needs at most 56 attempts (i.e. all possible rotations of the extracted 56 bits) to decrypt the transmitted ciphertext.

4.3 Experimental Results

In order to assess the effectiveness of the proposed post-deployment trust evaluation method, measurements are collected from multiple instances of the wireless cryptographic IC described. These measurements are then processed in silicon through an on-chip classifier implemented on the reconfigurable neural network experimentation platform.

4.3.1 Dataset Generation

Using Spice-level Monte-Carlo simulation with $\pm 7.5\,\%$ process variations on all circuit parameters, 1K chip instances of the Trojan-free circuit are generated. Similarly, 1K chip instances of the Type-I Trojan-infested circuit and 1K chip instances of the Type-II Trojan-infested circuit are also generated. For each of the Trojan-free chip instances, the transmission power when a logic ‘1’ is transmitted is measured. In addition, the measurements of the two current sensors are collected when a pre-selected 64-bit block (i.e. alternating 0s and 1s) is transmitted. The same measurements are also collected for the 1K Type-I Trojan-infested chips and 1K Type-II Trojan-infested chips, with the Trojan first dormant and then activated.

4.3.2 Observations

The following observations are made before further analysis of the collected dataset is performed:

- The transmission power profile of the Trojan-free chip-instances is indistinguishable from the transmission power profile of the Type-I Trojan-infested and Type-II Trojan infested chip instances with the Trojan *dormant*. This is demonstrated in Figs. 12a–c, where the transmission power is depicted for the chip instances of each of these three populations, enclosed within the $\pm 3\sigma$ boundary of the Trojan-free chip population. As may be observed, given any one of these transmission waveforms, it is impossible to definitively place it to one of the three populations. Even more interestingly, the transmission power profile

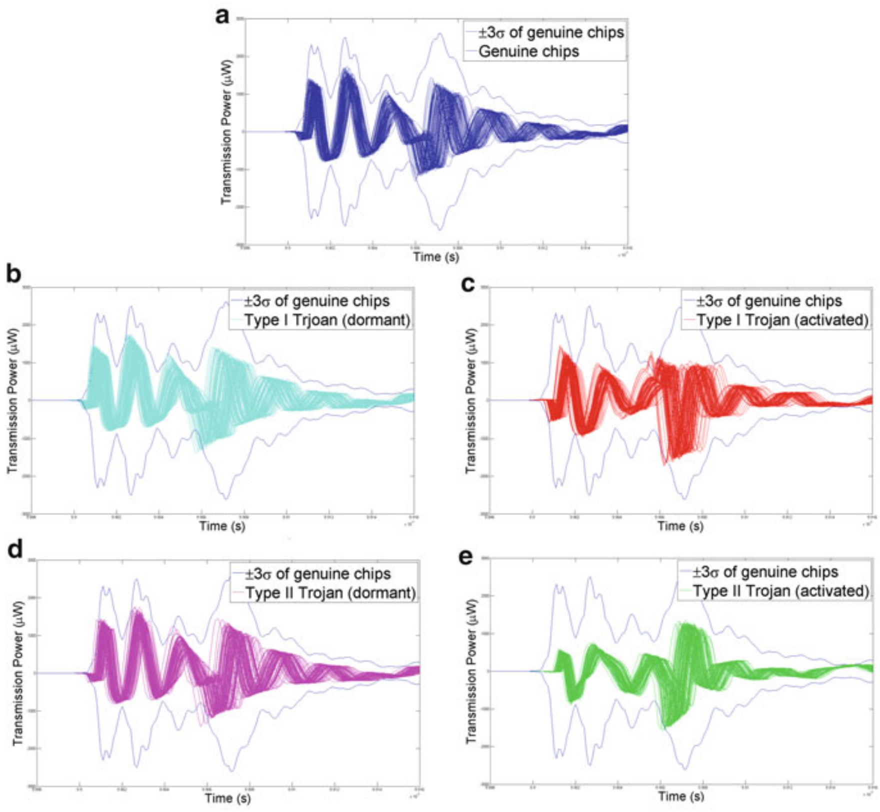


Fig. 12 3σ transmission power envelope of Trojan-free chip instances enclosing the various chip populations in the dataset. (**a**) Trojan-free chip instances. (**b**) Type-I Trojan-infested chip instances (Trojan Dormant). (**c**) Type-II Trojan-infested chip instances (Trojan Dormant) (**d**) Type-I Trojan-infested chip instances (Trojan Activated). (**e**) Type-I Trojan-infested chip instances (Trojan Activated)

of the Type-I Trojan-infested and Type-II Trojan infested chip instances with the Trojan *active* is also indistinguishable from the aforementioned populations, as shown in Figs. 12d–e. This is consistent with the results reported in [21] and affirms that the hardware Trojans do not violate the circuit specifications. In other words, a transmission of a Trojan-infested circuit with the Trojan activated appears to be perfectly legitimate and within the margins allowed for process variations and operational conditions fluctuation, hence the Trojans evade detection.

- The current sensor measurements of the Trojan-free chip instances are indistinguishable from the current sensor measurements of the Type-I Trojan-infested and Type-II Trojan infested chip instances with the Trojan *dormant*. This is demonstrated in Fig. 13 which depicts the three chip populations on the

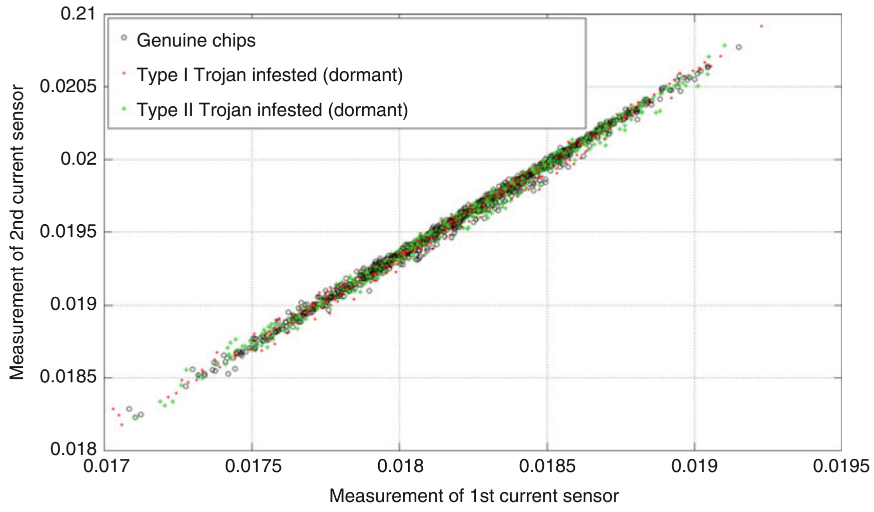


Fig. 13 Current sensor measurements with Trojans dormant

two-dimensional space of the current measurements. Evidently, the three populations fall upon each other, attesting to the inadequacy of pre-deployment methods in detecting dormant Trojans.

- The current sensor measurements of the Trojan-infested chip instances with the Trojan activated are distinguishable from the current sensor measurements of the Trojan-infested chip instances with the Trojan dormant. This is demonstrated in Figs. 14 and 15 for each of the two Trojan types. As may be observed, while each current sensor measurement by itself is insufficient to separate the Trojan-active and Trojan-dormant populations, their combination provides adequate information to do so. Therefore, it is possible that a trained on-chip classifier will be able to pick up the difference in the current sensor measurements when the Trojan is activated post-deployment and, thereby, alert of untrusted circuit operation, as aimed by the proposed methodology.

4.3.3 On-Chip Classifier Construction and Training

The reconfigurable neural network experimentation platform chip [28] described in Sect. 4.2 provides classifiers involving a range of neurons and various different topologies. In order to train an on-chip classifier to distinguish trusted from untrusted functionality, the testers should only rely on information from Trojan-free chips (or Trojan infested chips with the Trojan dormant, if Trojan-free chips are unavailable). This is important because, in a realistic scenario, the testers do not have advance knowledge of the various different types of Trojans and their potential impact, which will only appear after deployment of the chip. Therefore,

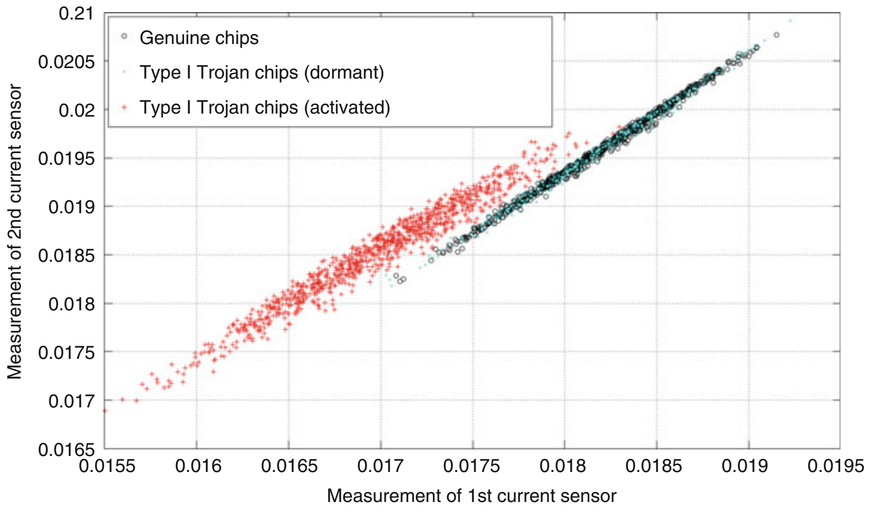


Fig. 14 Current sensor measurements for Type-I Trojan-infested chips

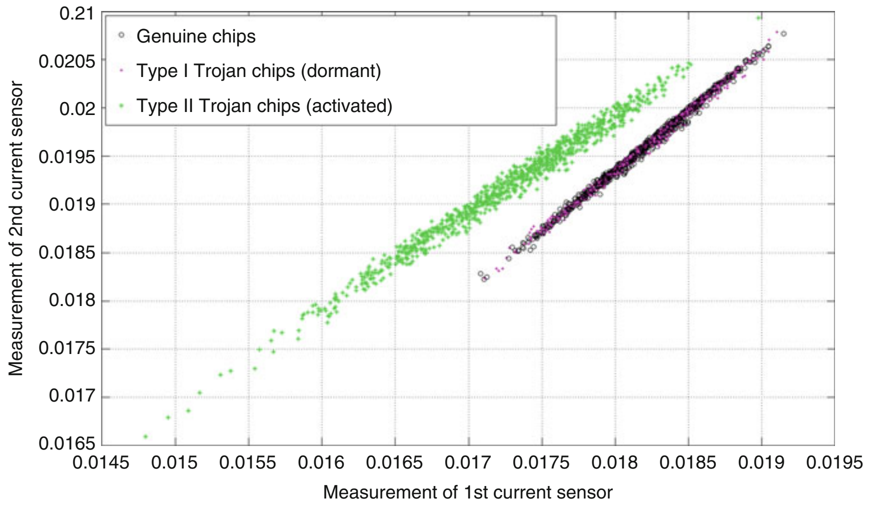


Fig. 15 Current sensor measurements for Type-II Trojan-infested chips

in the experiments only the data (i.e. the two current sensor measurements) from the 1K Trojan-free chip instances to train the classifier is used. In other words, it is a 1-class classification problem, where the objective is to train a classifier to enclose the region of acceptable (trusted) functionality without any data of unacceptable (untrusted) functionality. To this end, the 1-class classification training algorithm described in [37] is employed. As can be observed in Fig. 16, the boundary enclosing the trusted behavior is an ellipsoid, which can be approximated through a fairly

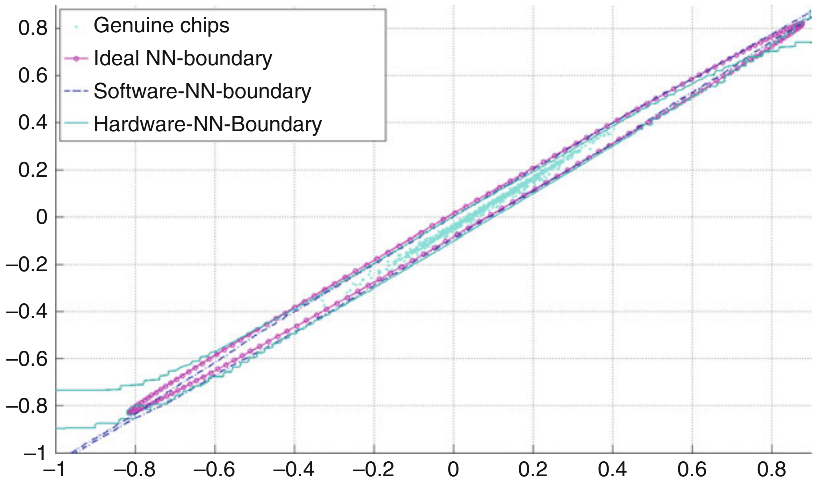


Fig. 16 The trained boundary learned through software NN and hardware NN for Trojan-free chips

simple two-layer neural network topology involving 4 neurons. The boundary shown in Fig. 16 is the actual boundary learned by the trained on-chip neural network. As a point of reference, the boundary learned by the software version of the selected neural network is also showed. Evidently, the boundary learned in hardware is essentially identical to the one learned in software.

4.3.4 On-Chip Trust Evaluating Effectiveness

After training, the on-chip classifier with the data from Trojan-free chip instances is assessed for its effectiveness in correctly classifying the two types of Trojan-infested chip populations. In order to obtain a global picture, the trained classifier is presented with the data from both when the Trojan is dormant and when the Trojan is activated. The former will allow the testers to evaluate the false positive rate (i.e. incorrectly rejecting a chip when the Trojan is dormant) and the false negative rate (i.e. incorrectly accepting a chip when the Trojan is active). Figure 17 depicts the learned boundary, along with the footprints of the Type-I Trojan-infested chip instances with the Trojan dormant and active. Similarly, Fig. 18 depicts the learned boundary, along with the footprints of the Type-II Trojan-infested chip instances with the Trojan dormant and active. As may be observed, the trained classifier performs extremely well and almost perfectly encapsulates the chip populations when the Trojan is dormant, while almost perfectly excluding the chip populations when the Trojan is activated. Tables 1 and 2 report the confusion matrices for the Type-I and Type-II Trojan-infested chip populations, respectively. For comparison, the effectiveness of the software version of the classifier is also reported, demonstrating that the error due to the hardware implementation is minimal.

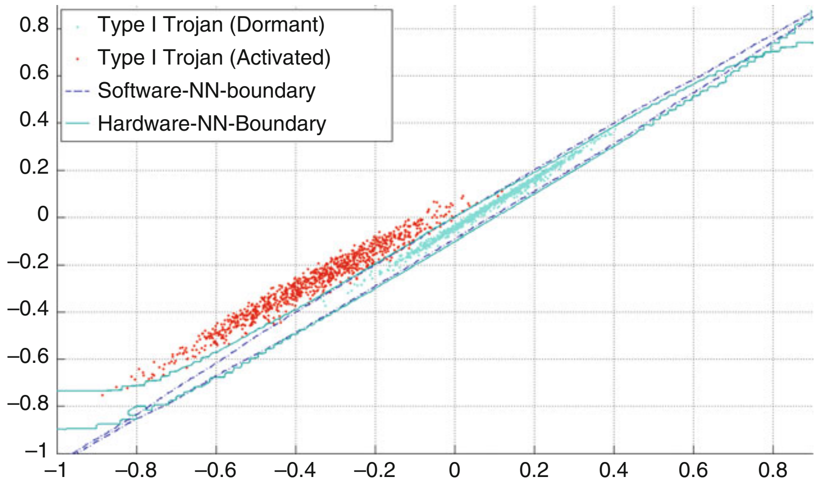


Fig. 17 Ability of boundary learned through software and hardware NN to correctly classify dormant and activated Type-I Trojan-infested chips

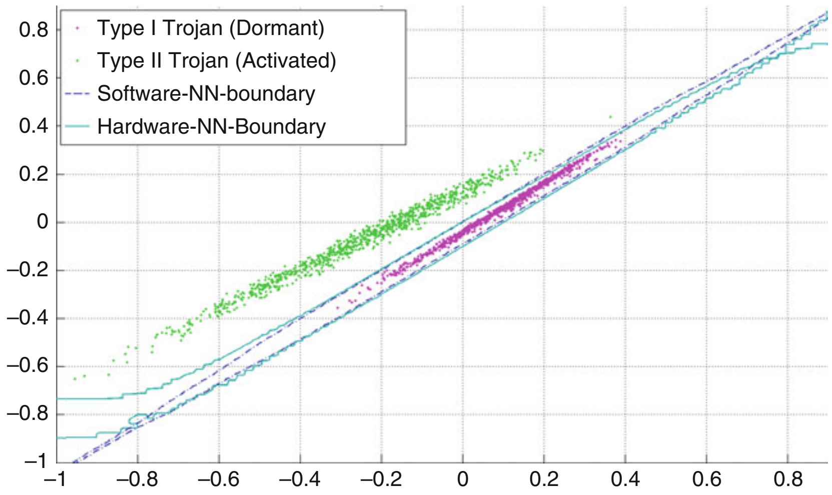


Fig. 18 Ability of boundary learned through software and hardware NN to correctly classify dormant and activated Type-II Trojan-infested chips

While not zero, the false positive and false negative rates are very low, indicating that the proposed on-chip classifier-based methodology has the potential of providing an effective post-deployment trust evaluation capability. Further research is still required towards achieving zero misclassification rate to ensure trustworthiness of deployed circuits.

Table 1 Type I Trojan classification

		Classified by hardware		Classified by software	
		Dormant	Activated	Dormant	Activated
Actual	Dormant	99.9 %	0.1 %	100 %	0 %
	Activated	2.8 %	97.2 %	1.3 %	98.7 %

Table 2 Type II Trojan classification

		Classified by hardware		Classified by software	
		Dormant	Activated	Dormant	Activated
Actual	Dormant	99.8 %	0.2 %	100 %	0 %
	Activated	0 %	100 %	0 %	100 %

5 Conclusion

The threat of hardware Trojans has fueled recent research in evaluating trustworthiness of fabricated ICs, both in the digital and in the analog/RF domains. In this chapter, current hardware Trojan detection methods in the analog/RF domain and, more specifically, in wireless cryptographic ICs was introduced. Using a Trojan-free and two Trojan-infested versions of a DES encryption core and a UWB transmitter, we demonstrated (i) the simplicity of a hardware Trojan attack and the ease with which it can leak sensitive information such as the encryption key, (ii) the inability of existing manufacturing test and hardware Trojan detection methods developed for digital circuit to detect such hardware Trojans in the analog/RF domain, and (iii) the power of side-channel fingerprinting in detecting such Trojans using statistical analysis of the transmission power waveforms and trained classifiers to distinguish between hardware Trojan-free an hardware Trojan-infested ICs. Furthermore, the problem of dormant hardware Trojans which are inactive during pre-deployment test and are only activated after deployment was discussed, along with the new challenges that it presents. Accordingly, a post-deployment trust evaluation architecture was introduced, wherein on-chip resources including sensors and a classifier are added to the IC, in order to support hardware Trojan detection in the field of operation. While these solutions provide an excellent initial step towards thwarting hardware Trojans in wireless cryptographic circuits, there remain plenty of challenges and opportunities for further research towards developing an arsenal of hardware Trojan prevention and detection methods for analog/RF ICs.

References

1. http://www.mosis.com/products/fab/vendors/tsmc/tsmc013-cl
2. http://www.opencores.org/projects.cgi/web/des/overview
3. Abdallah, L., Stratigopoulos, H.G., Kelma, C., Mir, S.: Sensors for built-in alternate rf test. In: IEEE 15th European Test Symposium (ETS) 2010, pp. 49–54 (2010)

4. Adee, S.: The hunt for the kill switch. IEEE Spectr. **45**(5), 34–39 (2008)
5. Agrawal, D., Baktir, S., Karakoyunlu, D., Rohatgi, P., Sunar, B.: Trojan detection using IC fingerprinting. In: IEEE Symposium on Security and Privacy, pp. 296–310 (2007)
6. Banga, M., Chandrasekar, M., Fang, L., Hsiao, M.S.: Guided test generation for isolation and detection of embedded Trojans in ICs. In: Proceedings of the 18th ACM Great Lakes symposium on VLSI, pp. 363–366 (2008)
7. Banga, M., Hsiao, M.: A novel sustained vector technique for the detection of hardware Trojans. In: Proceedings of the 22nd International Conference on VLSI Design, pp. 327–332 (2009)
8. Banga, M., Hsiao, M.: VITAMIN: Voltage inversion technique to asertain malicious insertion in ICs. In: IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 104–107 (2009)
9. Bloom, G., Narahari, B., Simha, R., Zambreno, J.: Providing secure execution environments with a last line of defense against Trojan circuit attacks. Comput. Secur. **28**(7), 660–669 (2009)
10. Bloom, G., Simha, R., Narahari, B.: OS support for detecting Trojan circuit attacks. In: IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 100–103 (2009)
11. Candore, A., Kocabas, O., Koushanfar, F.: Robust stable radiometric fingerprinting for frequency reconfigurable devices. In: IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 43–49 (2009)
12. Chakraborty, R., Wolff, F., Paul, S., Papachristou, C., Bhunia, S.: MERO: a statistical approach for hardware Trojan detection. In: Cryptographic hardware and embedded systems. Lecture Notes in Computer Science, vol. 5747, pp. 396–410 (2009)
13. Cimino, M., Lapuyade, H., De Matos, M., Taris, T., Deval, Y., Begueret, J.: A robust 130nm-cmos built-in current sensor dedicated to rf applications. In: Eleventh IEEE European Test Symposium 2006, ETS ’06, pp. 151–158 (2006)
14. Drzevitzky, S., Kastens, U., Platzner, M.: Proof-carrying hardware: Towards runtime verification of reconfigurable modules. In: International Conference on Reconfigurable Computing and FPGAs, pp. 189–194 (2009)
15. Drzevitzky, S., Platzner, M.: Achieving hardware security for reconfigurable systems on chip by a proof-carrying code approach. In: Proceedings of the 6th International Workshop on Reconfigurable Communication-centric Systems-on-Chip, pp. 1–8 (2011)
16. Hicks, M., Finnicum, M., King, S.T., Martin, M.M.K., Smith, J.M.: Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 159–172 (2010)
17. Jha, N., Gupta, S.: Testing of Digital Systems. Cambridge University Press, Cambridge (2003)
18. Jin, Y., Kupp, N., Makris, M.: DFTT: Design for Trojan test. In: IEEE International Conference on Electronics Circuits and Systems, pp. 1175–1178 (2010)
19. Jin, Y., Kupp, N., Makris, Y.: Experiences in hardware Trojan design and implementation. In: IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 50–57 (2009)
20. Jin, Y., Makris, Y.: Hardware Trojan detection using path delay fingerprint. In: IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 51–57 (2008)
21. Jin, Y., Makris, Y.: Hardware Trojans in wireless cryptographic ICs. IEEE Des. Test Comput. **27**, 26–35 (2010)
22. Jin, Y., Makris, Y.: Proof carrying-based information flow tracking for data secrecy protection and hardware trust. In: IEEE 30th VLSI Test Symposium (VTS), pp. 252–257 (2012)
23. Jin, Y., Maliuk, D., Makris, Y.: Post-deployment trust evaluation in wireless cryptographic ICs. In: Design, Automation Test in Europe Conference (DATE), pp. 965–970 (2012)
24. Jin, Y., Yang, B., Makris, Y.: Cycle-accurate information assurance by proof-carrying based signal sensitivity tracing. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 99–106 (2013)
25. Lin, L., Burleson, W., Paar, C.: MOLES: Malicious off-chip leakage enabled by side-channels. In: Proceedings of the 2009 International Conference on Computer-Aided Design, ICCAD ’09, pp. 117–122. ACM, New York (2009)

26. Lin, L., Kasper, M., Guneysu, T., Paar, C., Burleson, W.: Trojan side-channels: lightweight hardware Trojans through side-channel engineering. In: Cryptographic Hardware and Embedded Systems, LNCS, vol. 5747, pp. 382–395. Springer, Berlin (2009)
27. Love, E., Jin, Y., Makris, Y.: Proof-carrying hardware intellectual property: a pathway to trusted module acquisition. IEEE Trans. Inf. Forensics Secur. **7**(1), 25–40 (2012)
28. Maliuk, D., Makris, Y.: A dual-mode weight storage analog neural network platform for on-chip applications. In: IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2889–2892 (2012)
29. Maliuk, D., Stratigopoulos, H., Huang, H., Makris, Y.: Analog neural network design for RF built-in self-test. In: Proceedings of the IEEE International Test Conference (ITC), pp. 23.2.1–23.2.10 (2010)
30. Moshtagh, N.: Minimum volume enclosing ellipsoid. In: GRASP Laboratory, University of Pennsylvania. http://www.seas.upenn.edu/~nima/papers/Mim_vol_ellipse.pdf(2005)
31. Nelson, M., Nahapetian, A., Koushanfar, F., Potkonjak, M.: SVD-based ghost circuitry detection. In: Information hiding. Lecture Notes in Computer Science, vol. 5806, pp. 221–234 (2009)
32. Potkonjak, M., Nahapetian, A., Nelson, M., Massey, T.: Hardware Trojan horse detection using gate-level characterization. In: Proceedings of the 46th Annual Design Automation Conference, DAC '09, pp. 688–693 (2009)
33. Rad, R., Plusquellic, J., Tehranipoor, M.: Sensitivity analysis to hardware Trojans using power supply transient signals. In: IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 3–7 (2008)
34. Rad, R.M., Wang, X., Tehranipoor, M., Plusquellic, J.: Power supply signal calibration techniques for improving detection resolution to hardware Trojans. In: IEEE/ACM International Conference on Computer-Aided Design, pp. 632–639 (2008)
35. Salmani, H., Tehranipoor, M., Plusquellic, J.: New design strategy for improving hardware Trojan detection and reducing Trojan activation time. In: IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 66–73 (2009)
36. Sinanoglu, O., Karimi, N., Rajendran, J., Karri, R., Jin, Y., Huang, K., Makris, Y.: Reconciling the IC test and security dichotomy. In: Proceedings of the 18th IEEE European Test Symposium (ETS), pp. 1–6 (2013)
37. Skabar, A.: Single-class classifier learning using neural networks: an application to the prediction of mineral deposits. In: The 2nd International Conference on Machine Learning and Cybernetics, pp. 2127–2132 (2003)
38. Stratigopoulos, H.G., Makris, Y.: Error moderation in low-cost machine-learning-based analog/RF testing. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **27**(2), 339–351 (2008)
39. Waksman, A., Suozzo, M., Sethumadhavan, S.: FANCI: Identification of stealthy malicious logic using boolean functional analysis. In: Proceedings of the ACM SIGSAC Conference on Computer & Communications Security, CCS '13, pp. 697–708 (2013)
40. Wolff, F., Papachristou, C., Bhunia, S., Chakraborty, R.S.: Towards Trojan-free trusted ICs: Problem analysis and detection scheme. In: IEEE Design Automation and Test in Europe, pp. 1362–1365 (2008)
41. Yuan, T., Zheng, Y., Ang, C., Li, L.: A fully integrated CMOS transmitter for ultra-wideband applications. In: IEEE Radio Frequency Integrated Circuits Symposium, pp. 39–42 (2007)

Obfuscation-Based Secure SoC Design for Protection Against Piracy and Trojan Attacks

Rajat Subhra Chakraborty, Yu Zheng, and Swarup Bhunia

Abstract System-on-Chip (SoC) designs rely heavily on reusable and pre-verified hardware intellectual property (IP) cores. Recent trends of IP piracy and reverse-engineering, and malicious circuit modifications ("hardware Trojans") in remote fabrication facilities, are major concerns. In this chapter, we propose a comprehensive secure SoC design flow based on the principle of *design obfuscation* that can provide effective protection against IP piracy as well as hardware Trojan attacks. We present the overall approach and illustrate the obfuscation process with an example. We provide a brief survey of related work on hardware obfuscation and then present two variations of obfuscation, which differ in the level of protection and the complexity of implementation. The first approach is based on extraction and modification of the state transition from the gate-level synthesized design, such that normal operation is possible only on the successful application of a correct initialization sequence. Optionally, an obfuscated register transfer level (RTL) design can be generated by de-compilation of the obfuscated netlist. In the second approach, a register transfer level IP is obfuscated by manipulating its control and data flow graphs. The proposed approaches are scalable, and can be integrated in the SoC design and manufacturing flow to benefit all parties associated with the design flow, while minimally affecting the end-user experience.

1 Introduction

Reuse-based SoC design using hardware IP cores has become a pervasive practice in the industry. These IP cores usually come in the following three forms: synthesizable Register Transfer Level (RTL) descriptions in Hardware Description Languages

R.S. Chakraborty (✉)
Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, West Bengal 721302, India
e-mail: rschakraborty@cse.iitkgp.ernet.in

Y. Zheng • S. Bhunia
Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH 44106, USA
e-mail: yxz402@case.edu; skb21@case.edu

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_8

(HDLs) ("Soft IP"); gate-level designs directly implementable in hardware ("Firm IP"); and GDS-II design database ("Hard IP"). The approach of designing complex systems by integrating tested, verified and reusable modules reduces the SoC design time and cost dramatically [8].

Unfortunately, recent trends in IP-piracy and reverse-engineering efforts to produce counterfeit ICs have raised serious concerns in the IC design community [8, 15, 28, 29, 33]. IP piracy can take diverse forms, as illustrated by the following scenarios:

- A chip design house buys the IP core from the IP vendor, and makes an illegal copy or "clone" of the IP. The IC design house then uses the IP without paying the required royalty, or sells it to another IC design house (after minor modifications) claiming the IP to be its own design [28].
- An untrusted fabrication house makes an illegal copy of the GDS-II database supplied by a chip design house, and then manufactures and sells counterfeit copies of the IC under a different brand name [9].
- A company performs post-silicon reverse-engineering on an IC to manufacture its illegal clone [38].

These scenarios demonstrate that all parties involved in the IC design flow are vulnerable to different forms of IP infringement which can result in loss of revenue and market share. *Obfuscation* is a technique that transforms an application or a design into one that is functionally equivalent to the original but is significantly more difficult to reverse engineer. Software obfuscation to prevent reverse-engineering has been studied widely in recent years [17, 18, 22, 23, 32, 42]; however, the techniques of software obfuscation are not directly applicable to HDL because the obfuscated HDL can result in potentially unacceptable design overhead when synthesized.

Although design modifications to prevent the illegal manufacturing of ICs by fabrication houses have been proposed [38] before, such techniques are not useful in preventing the theft of soft IPs. Furthermore, they do not provide protection against possible IP piracy from the SoC design house. In this chapter, we present two low-overhead techniques each of which can serve as the basis for a secure SoC design methodology through design obfuscation and authentication performed on the RTL design description. We follow a key-based obfuscation approach, where normal functionality is enabled only upon application of a specific input initialization key sequence. Majority of the commercial hardware IPs come in the RTL ("soft") format which offers better portability by allowing design houses to map the circuit to a preferred platform in a particular manufacturing process [16]. Recently, some authors have re-phrased the concept of functional obfuscation as *logic encryption* [35]. The motivation behind this is the possibility of confusing the concepts of hardware obfuscation with the way the term "obfuscation" is used in the software context. In software obfuscation, the functionality of the obfuscated program is preserved, although its form is modified. However, this is not the case in the hardware obfuscation case. However, in this chapter, we retain the "hardware obfuscation" terminology, to retain continuity with our previous work.

We have previously proposed design obfuscation for gate-level IPs [9]. The main contributions of the work presented in that paper are: (a) finding the optimal modification locations in a gate-level netlist by structural analysis, and, (b) designing a gate-level modification circuit to optimally modify the netlist. However, the work did not perform analysis of the actual functionality of the circuit in terms of its data and control flow patterns. While direct structural analysis and the structural modification proposed in [9] is suitable for gate-level IPs, the proposed obfuscation approach for gate-level IPs cannot be used for RTL IPs for the following reasons: (a) RTL IPs does not directly provide high-level structural information; and (b) any obfuscation performed on RTL IP should maintain its portability. Since a majority of the hardware IPs are actually delivered in RTL format, there is a need to develop low-overhead protection measures for them against piracy. In this chapter, we propose a anti-piracy approach for RTL IPs based on effective key-based hardware obfuscation. The basic idea is to judiciously obfuscate the control and data flow of an RTL IP in a way that prevents its normal mode operation without application of an enabling key at its primary input. To the best of our knowledge, this is the first work that provides RTL hardware IP protection through key-based obfuscation. In particular, we make the following contributions in this work:

- We provide two RTL IP obfuscation solutions, which differ in level of protection and computation complexity: 1) the first technique, referred to as the "STG modification approach", converts a RTL description to gate level; obfuscates the gate level netlist; and then de-compiles back to RTL; 2) the second technique, referred to as the "CDFG modification approach", avoids the forward compilation step and applies obfuscation in the register transfer level by modifying its control and data flow constructs, which is facilitated through generation of a CDFG of the RTL design. The first approach can provide higher level of obfuscation but is computationally more expensive than the second one.
- We derive appropriate metrics to quantify the obfuscation level for both the approaches. We compare the two approaches both qualitatively and quantitatively. We show that an important advantage of the proposed obfuscation approach is that the level of protection can be improved with minimal additional hardware overhead by increasing the length of the initialization key sequence. We show that this is unlike conventional encryption algorithms e.g. the *Advanced Encryption Standard* (AES) whose hardware implementations usually incur much larger overhead for increasing key length.
- Finally, along with obfuscation, we show that the proposed approaches can be extended to embed a hard-to-remove "digital watermark" in the IP that can help to authenticate the IP in case it is illegally stolen. The authentication capability provides enhanced security while incurring little additional hardware overhead. Also, we show that by utilizing the automatic propagation of the security features to lower levels of design abstraction, we can provide protection against *hardware Trojan* insertion in fabrication facilities [13], following the principles outlined in our previous research [10].

The rest of the chapter is organized as follows. In Sect. 2, we present related work and the motivation behind this work. In Sect. 3, we describe the two IP obfuscation schemes, and their relative merits and demerits. In Sect. 4, we present theoretical analysis of the proposed obfuscation schemes to derive metrics for the level of obfuscation. In Sect. 5 we present automated design flows, and simulation results for several open-source IP cores. In Sect. 6, we describe a technique to decrease the overhead by utilizing the normally unused states of the circuit. We conclude in Sect. 7.

2 Related Work

Hardware IP protection has been investigated earlier in diverse contexts. Previous work on this topic can be broadly classified into two main categories: (1) *Obfuscation* based protection, and (2) *Authentication* based protection.

Hardware Obfuscation In obfuscation based IP protection, the IP vendor usually affects the human readability of the HDL code [40], or relies on cryptographic techniques to encrypt the source code [31]. In [40], the code is re-formatted by changing the internal net names and removing the comments, so that the circuit description is no longer intelligible to the human reader. RTL obfuscation for VHDL descriptions has been explored in [7], where the authors use rudimentary transformations of the code such as variable name changes, inlining of code, loop unrolling, statement order changing, etc. to make the code difficult to understand. However, usually the IP interface and port names cannot be modified or obfuscated to comply with the specifications. As the above two techniques do not modify the functionality of the IP core, they cannot prevent an IP from being stolen by an adversary and used as a "black-box" circuit module. In [31], the HDL source code is encrypted and the IP vendor provides the key to de-crypt the source code only to its customers using a particular secure design platform. A similar approach has been proposed in [41], where an infrastructure for IP evaluation and delivery for FPGA applications has been proposed based on Java applets. However, the above techniques force the use of a particular design platform, a situation that might be unacceptable to many SoC designers who seek the flexibility of multiple tools from diverse vendors in the design flow. Also, none of the above techniques prevent possible reverse-engineering effort at later stages of the design and manufacturing flow (to produce IC clones or insert hardware Trojan), and thus does not benefit the other associated parties (e.g. the SoC designers and the system designers). One important point to note is that a given hardware obfuscation technique should not affect any change in the standard interface of a hardware IP module. In other words, obfuscation of a hardware IP module should have minimum impact on the workflow of an IC designer.

Hardware Watermarking To protect the rights of the IP vendor through authentication, the approaches proposed are directed towards embedding a well-hidden *Digital*

Watermark in the design [8, 15, 28, 29, 33] to authenticate the design at a later stage. Since this digital watermark (or signature) is extremely difficult to discover and remove from the IP, it is easy to prove an illegal use of such a component in litigation. The signature is typically in the form of an input–output response pair hosted in the memory or combinational parts of the circuit [8, 33], but secure hashing to insert multiple small watermarks in the design has also been proposed [29]. Another noted approach is *constraint-based watermarking* [28], where the watermarks are design features which result from constraints applied to optimization and constraint-satisfaction problems during design. The authentication-based IP protection schemes are efficient and sophisticated techniques to prove the ownership of an IP which has a digital signature embedded in it. However, their main shortcoming is they are *passive*, in the sense that they cannot prevent the stolen IP from being used in the first place. They can only help to prove the ownership in case of litigation.

Software Obfuscation It has been shown that it is possible to recover software source-code by "de-compilation" of binary or byte-code. One of the most popular approaches of preventing such reverse-engineering is to obfuscate the control-flow of a program to produce "spaghetti code" that is difficult to de-compile from the binary form to a higher level program [22, 23, 30, 42]. Another technique is the so-called "code morphing" [32], where a section of the compiled code is substituted with an entirely new block that expects the same machine state when it begins execution of the previous section, and leaves with the same machine state after execution as the original. Other software obfuscation approaches include self-modifying code [26] (code that generates other code at run-time), self-decryption of partially encrypted code at run-time [3, 39], and code redundancy and voting to produce "tamper-tolerant software" (conceptually similar to hardware redundancy for fault tolerance) [25]. A general shortcoming of these approaches is that they do not scale well in terms of memory footprint and performance as the size of the program (or the part of the program to be protected) increases [14]. Hence, RTL obfuscation approaches motivated along similar lines are also likely to result in inefficient circuit implementations of the obfuscated RTL with unacceptable design overhead. Also, the value of such techniques is the matter of debate because it has been theoretically proven that software obfuscation in terms of obfuscating the "black-box functionality" does not exist [4]. In contrast, we modify both the structure and the functionality of the circuit description under question; hence the above result of impossibility of obfuscation is not applicable in our case.

3 Obfuscation Methodology

In this section, we describe the two proposed obfuscation-based RTL IP protection schemes. Figure 1 compares the major steps of the proposed schemes. We also describe the enhanced design flow that we have developed by integrating them with the traditional SoC design and manufacturing flow [9]. First, we describe the STG

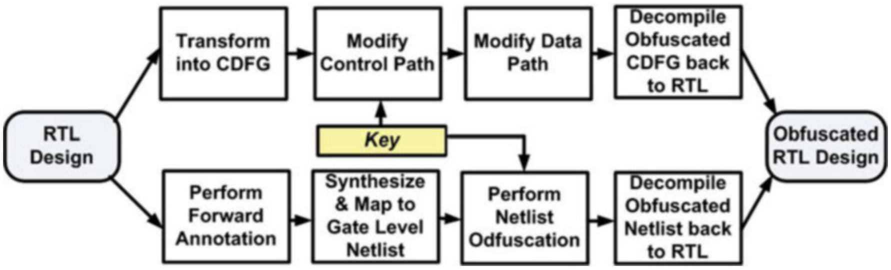

RTL Design
Transform into CDFG
Modify Control Path
Modify Data Path
Decompile Obfuscated CDFG back to RTL
Key
Perform Forward Annotation
Synthesize & Map to Gate Level Netlist
Perform Netlist Odfuscation
Decompile Obfuscated Netlist back to RTL
Obfuscated RTL Design

Fig. 1 Comparison of the two proposed schemes

modification and de-compilation based IP protection technique [11], followed by the technique based on CDFG modification [12]. We end the section with a comparison of the relative advantages and disadvantages of the two approaches.

3.1 STG Modification Approach

3.1.1 Methodology

This technique has three main steps:

- Logic synthesis of the RTL to an unmapped, unoptimized gate-level netlist, composed of generic Boolean gates.
- Functional obfuscation of the netlist by structural modifications following the principle outlined in [9], and,
- Subsequent de-compilation of the obfuscated netlist back to an obfuscated version of the original RTL.

The advantage of modifying the functionality at the gate-level is the relative incomprehensibility of such a circuit description compared to a RTL description. The goal is to modify the STG of the circuit, such that on power-up, the circuit operates in an *obfuscated mode* when its functionality is incorrect. Figure 2a shows the modification of the STG of a circuit, and Fig. 2b shows the structural modifications that achieve this. Only on the correct application of an *initialization key sequence* ($P0 \rightarrow P1 \rightarrow P2$ in Fig. 2a), the circuit goes to the *normal mode* when the original functionality is restored; otherwise, the circuit retains its incorrect functionality. The obfuscation is achieved following a scheme described in [9], whereby selected internal nodes in the circuit are modified using special *modification cells*. A *mode control* Finite State Machine (FSM) is integrated in the circuit which generates certain control signals to enable or disable the modification cells. The modification cells change the Boolean functionality of a node from f to f_{mod} in the following way:

$$f_{mod} = f\cdot\overline{en} + \overline{f}\cdot g\cdot en \qquad (1)$$

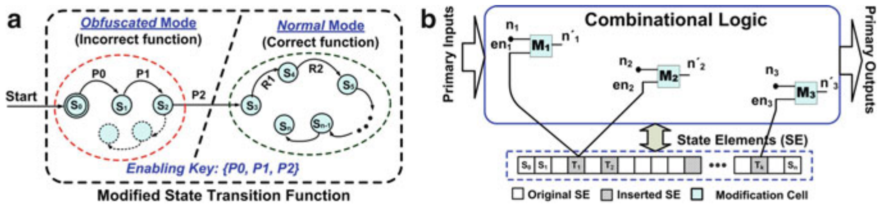
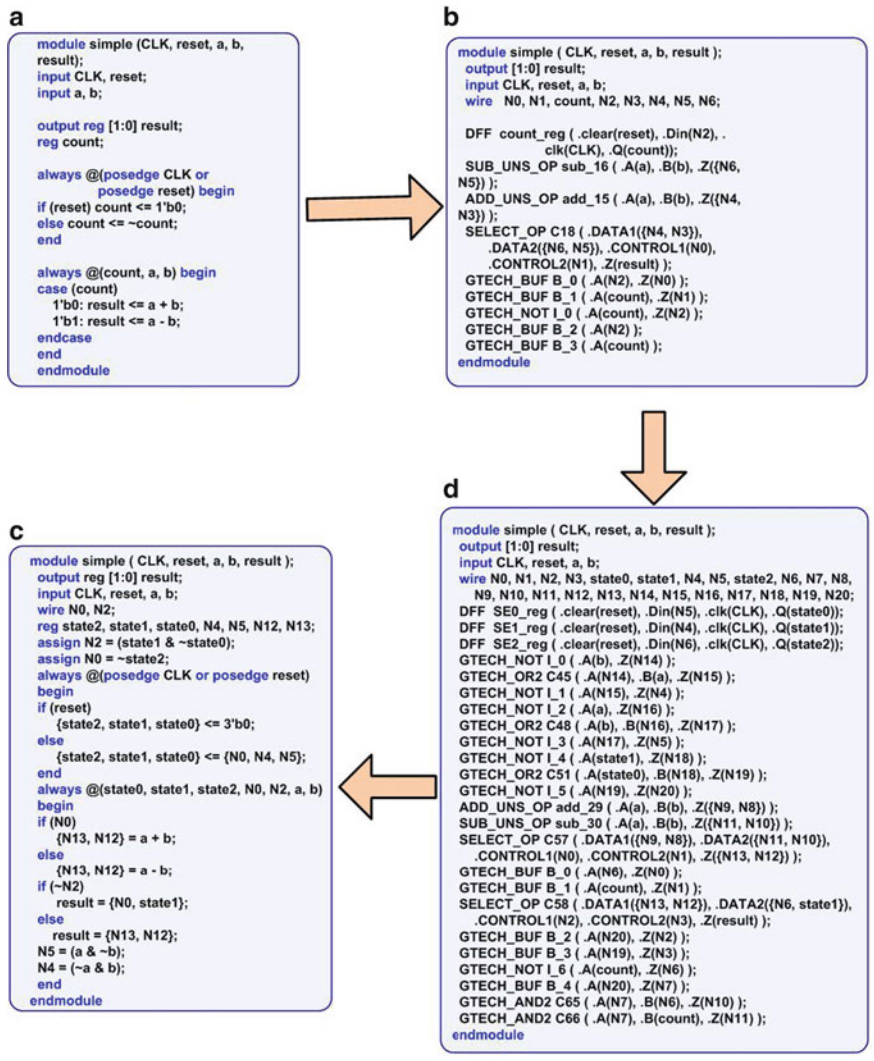


Fig. 2 The functional obfuscation scheme by state transition graph (STG) modification [9, 11]: (**a**) modified state transition function; and (**b**) change in internal node structure using modification cells. Here, on power-on, the circuit is in the *obfuscated mode*, and on application of a correct *initialization key sequence* of $P_0 \rightarrow P_1 \rightarrow P_2$, the circuit goes to the *normal mode*

where the signal *en* is a mode-control signal derived from the inserted *mode control FSM* and *g* is another Boolean logic function called the “Modification Kernel Function” (MKF) that helps to increase the dissimilarity between f and f_{mod} [9].

The selection of the nodes and the design of the *modification cell* ensures maximum perturbation of a design for a given total number of modifications, as determined by the hardware overhead constraint. After the netlist modifications have been performed, the entire circuit is re-synthesized to “flatten” the modifications to a single netlist. This process allows logic sharing between the original circuit, the inserted FSM and the *modification cells*, thus helping to make the circuit modifications unidentifiable. This is indicated in Fig. 2b by the intermingled original state elements and the state elements of the inserted FSM. The modified gate-level design is then de-compiled to regenerate the RTL of the code, without maintaining high level HDL constructs. Instead, the modified netlist is traversed recursively to reconstruct the Boolean equations for the primary output nodes and the state element inputs, expressed in terms of the primary inputs, the state-element outputs and a few selected high fanout internal nodes. The redundant internal nodes are then removed. This “partial flattening” effect hides all information about the modifications performed in the netlist. Optionally, the obfuscation tool maintains a list of expected instances of library datapath elements, and whenever these are encountered in the netlist, their outputs are related through proper RTL constructs to their inputs. This ensures regeneration of the same datapath cells on resynthesis of the RTL.

As an example, consider the simple Verilog module “*simple*” which performs addition or subtraction of two bits depending on the value of a free running one-bit counter, as shown in Fig. 3a. Figure 3b–d shows the transformation of the design through the proposed obfuscation process. The de-compiled RTL in Fig. 3d shows that the modification cell and the extra state transition logic are effectively hidden and isolation of the correct initialization sequence can be difficult even for such a small design. Major semantic effect of obfuscation is the change and replacement of high level RTL constructs (such as `if...else`, `for`, `while`, `case`, `assign` etc.) in the original RTL, and replacement of internal nodes and registers. Furthermore, internal register, net and instance names are changed to arbitrary identifiers to make the code less comprehensible.



a
```
module simple (CLK, reset, a, b,
result);
input CLK, reset;
input a, b;

output reg [1:0] result;
reg count;

always @(posedge CLK or
          posedge reset) begin
if (reset) count <= 1'b0;
else count <= ~count;
end

always @(count, a, b) begin
case (count)
  1'b0: result <= a + b;
  1'b1: result <= a - b;
endcase
end
endmodule
```

b
```
module simple ( CLK, reset, a, b, result );
  output [1:0] result;
  input CLK, reset, a, b;
  wire  N0, N1, count, N2, N3, N4, N5, N6;

  DFF count_reg ( .clear(reset), .Din(N2), .
              clk(CLK), .Q(count));
  SUB_UNS_OP sub_16 ( .A(a), .B(b), .Z({N6,
N5}) );
  ADD_UNS_OP add_15 ( .A(a), .B(b), .Z({N4,
N3}) );
  SELECT_OP C18 ( .DATA1({N4, N3}),
     .DATA2({N6, N5}), .CONTROL1(N0),
     .CONTROL2(N1), .Z(result) );
  GTECH_BUF B_0 ( .A(N2), .Z(N0) );
  GTECH_BUF B_1 ( .A(count), .Z(N1) );
  GTECH_NOT I_0 ( .A(count), .Z(N2) );
  GTECH_BUF B_2 ( .A(N2) );
  GTECH_BUF B_3 ( .A(count) );
endmodule
```

d
```
module simple ( CLK, reset, a, b, result );
 output [1:0] result;
 input CLK, reset, a, b;
 wire N0, N1, N2, N3, state0, state1, N4, N5, state2, N6, N7, N8,
    N9, N10, N11, N12, N13, N14, N15, N16, N17, N18, N19, N20;
 DFF  SE0_reg ( .clear(reset), .Din(N5), .clk(CLK), .Q(state0));
 DFF  SE1_reg ( .clear(reset), .Din(N4), .clk(CLK), .Q(state1));
 DFF  SE2_reg ( .clear(reset), .Din(N6), .clk(CLK), .Q(state2));
 GTECH_NOT I_0 ( .A(b), .Z(N14) );
 GTECH_OR2 C45 ( .A(N14), .B(a), .Z(N15) );
 GTECH_NOT I_1 ( .A(N15), .Z(N4) );
 GTECH_NOT I_2 ( .A(a), .Z(N16) );
 GTECH_OR2 C48 ( .A(b), .B(N16), .Z(N17) );
 GTECH_NOT I_3 ( .A(N17), .Z(N5) );
 GTECH_NOT I_4 ( .A(state1), .Z(N18) );
 GTECH_OR2 C51 ( .A(state0), .B(N18), .Z(N19) );
 GTECH_NOT I_5 ( .A(N19), .Z(N20) );
 ADD_UNS_OP add_29 ( .A(a), .B(b), .Z({N9, N8}) );
 SUB_UNS_OP sub_30 ( .A(a), .B(b), .Z({N11, N10}) );
 SELECT_OP C57 ( .DATA1({N9, N8}), .DATA2({N11, N10}),
   .CONTROL1(N0), .CONTROL2(N1), .Z({N13, N12}) );
 GTECH_BUF B_0 ( .A(N6), .Z(N0) );
 GTECH_BUF B_1 ( .A(count), .Z(N1) );
 SELECT_OP C58 ( .DATA1({N13, N12}), .DATA2({N6, state1}),
   .CONTROL1(N2), .CONTROL2(N3), .Z(result) );
 GTECH_BUF B_2 ( .A(N20), .Z(N2) );
 GTECH_BUF B_3 ( .A(N19), .Z(N3) );
 GTECH_NOT I_6 ( .A(count), .Z(N6) );
 GTECH_BUF B_4 ( .A(N20), .Z(N7) );
 GTECH_AND2 C65 ( .A(N7), .B(N6), .Z(N10) );
 GTECH_AND2 C66 ( .A(N7), .B(count), .Z(N11) );
endmodule
```

c
```
module simple ( CLK, reset, a, b, result );
  output reg [1:0] result;
  input CLK, reset, a, b;
  wire N0, N2;
  reg state2, state1, state0, N4, N5, N12, N13;
  assign N2 = (state1 & ~state0);
  assign N0 = ~state2;
  always @(posedge CLK or posedge reset)
  begin
  if (reset)
     {state2, state1, state0} <= 3'b0;
  else
     {state2, state1, state0} <= {N0, N4, N5};
  end
  always @(state0, state1, state2, N0, N2, a, b)
  begin
  if (N0)
     {N13, N12} = a + b;
  else
     {N13, N12} = a - b;
  if (~N2)
     result = {N0, state1};
  else
    result = {N13, N12};
  N5 = (a & ~b);
  N4 = (~a & b);
  end
endmodule
```

Fig. 3 Example of a Verilog RTL description and its obfuscated version [11]: (**a**) original RTL; (**b**) technology independent, unoptimized gate-level netlist obtained through RTL compilation; (**c**) obfuscated gate-level netlist; (**d**) de-compiled obfuscated RTL

After the gate-level modification, the modified netlist is de-compiled to produce a description of the circuit, which although being technically a RTL and functionally equivalent to the modified gate-level netlist, is extremely difficult to comprehend to a human reader. In addition, the modifications made to the original circuit remain well-hidden. A forward annotation file indicates relevant high-level HDL constructs and macros to be preserved through this transformation. These are maintained

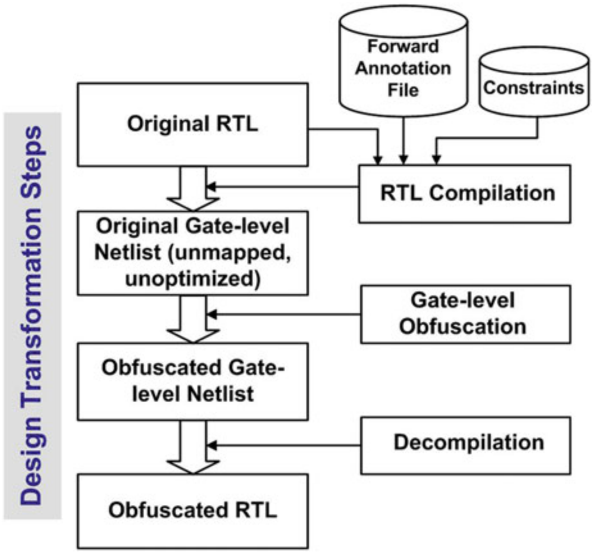


Fig. 4 Design transformation steps in course of the proposed RTL obfuscation process

during the RTL compilation and de-compilation steps. From the unmapped gate-level netlist, we look for specific generic gates, that can be de-compiled to an equivalent RTL construct, e.g. multiplexor can be mapped to an equivalent `if...then...else` construct or a `case` construct. The datapath modules or macros are transformed into appropriate operands.

For example, an equation $n1 = s1{\cdot}d1 + s2{\cdot}d2 + s3{\cdot}d3$ can be mapped to a `case` construct. Figure 4 shows the design transformation steps during the obfuscation process. We present an analysis of the security of this scheme in Sect. 4.

3.1.2 Embedding Authentication Features

To prevent against attacks from parties with knowledge of the initialization sequence in the design flow, the designer can optionally embed a *signature* within the design which acts as a *digital watermark*. In [8], the authors classify hardware watermarking schemes for authentication into four classes (in order of increasing security level): (a) physical-level watermarks; (b) synthesis-level watermarks; (c) high-level watermarks, and (d) multiple abstraction level watermarks. Although multiple level watermarks are the most robust, their implementation is complex and very costly [8]. Hence, in this work we go for the next option which is embedding of authentication features at high-level (HDL-level) design description.

This can be done by another modification of the state transition function, as shown in Fig. 5 [9]. Here, starting from the state S_0^O on power-on, on the application of an input sequence $P_1^A \rightarrow P_2^A \rightarrow P_3^A$, the design goes through a state transition sequence $S_0^A \rightarrow S_1^A \rightarrow S_2^A$. Corresponding to each of these states, a particular bit pattern is made to appear at the primary output. The state encoding for the states S_0^A

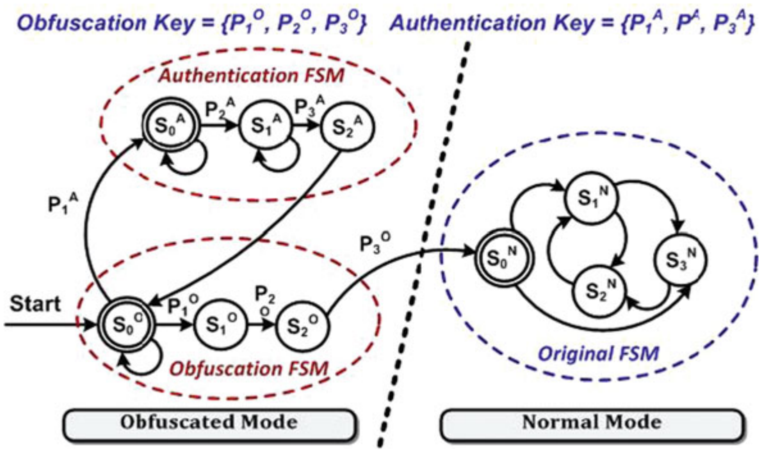


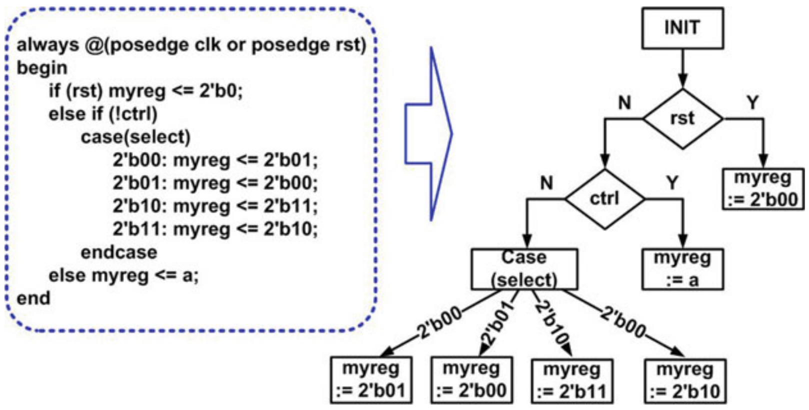
Fig. 5 Embedding authentication features in the circuit [9]

through S_2^A corresponds to *unused states* in the *normal mode* of operation, thereby guaranteeing that the above state transitions are not part of the original design, and is known only to the IP vendor. Hence, these states and the corresponding output act as the embedded digital watermark. Even if the adversary arranges to by-pass the initialization stage by structural modifications, because of the prevalent widespread use of full-scan designs, the inserted FSM flip-flops can always be controlled to have the desired bit-patterns corresponding to the states in the *authentication FSM*, thus revealing the watermark. As shown by the analysis in [9], this watermarking scheme is highly secure, with the probability of discovering the watermark in a circuit with 30 existing flip-flops and 3 flip-flops being used to encode the states in the authenticating mode is about $\sim 10^{-47}$. If all the reachable states can be derived by analyzing the RTL, then encoding the *authentication FSM* is easy; otherwise, *unused states* are to be derived from the gate-level netlist. Finding unreachable states for a large circuit with many state elements is computationally challenging; however it has been shown in [9] that by considering small groups of state elements in a given circuit and performing *sequential justification* using commercially available EDA tools, it is feasible to derive unreachable states for large circuits in reasonable time.

3.2 CDFG Modification Approach

3.2.1 Methodology

Similar to the STG modification based scheme, the main idea of this approach is to efficiently integrate a *mode control FSM* into the design through judicious modification of control and data flow structures derived from the RTL, such that



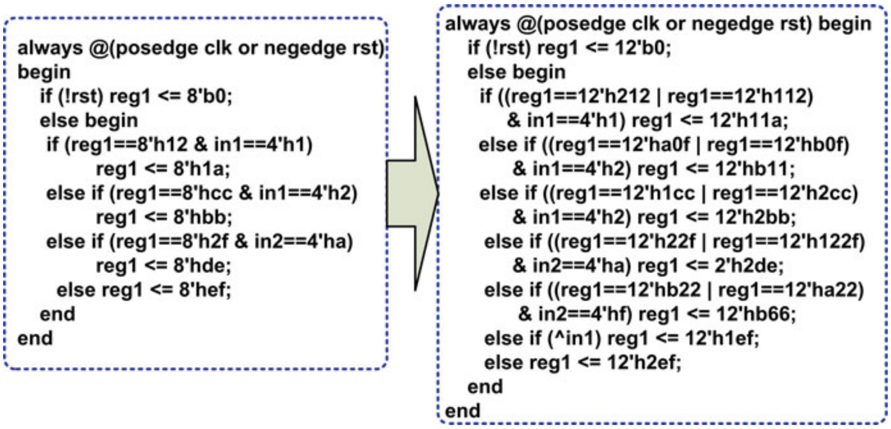
```
always @(posedge clk or posedge rst)
begin
    if (rst) myreg <= 2'b0;
    else if (!ctrl)
        case(select)
            2'b00: myreg <= 2'b01;
            2'b01: myreg <= 2'b00;
            2'b10: myreg <= 2'b11;
            2'b11: myreg <= 2'b10;
        endcase
    else myreg <= a;
end
```

Fig. 6 Transformation of a block of RTL code into CDFG [12]

the design works in two different modes *obfuscated* and *normal*. The operations of the *mode control FSM* is the same as that depicted in Fig. 2, with the circuit starting in the *obfuscated mode* and then moving to the *normal mode* only after the successful application of a pre-defined input sequence. The *mode control FSM* is integrated inside the CDFG derived from the RTL in a way that makes it extremely hard to isolate from the original IP. The FSM is realized in the RTL by expanding a judiciously selected set of registers, which we refer to as *host registers* and modifying their assignment conditions and values. Once the FSM has been integrated, both control and data flow statements are conditioned based on the mode control signals derived from this FSM. The proposed obfuscation scheme comprises of four major steps described below.

Parsing the RTL and Building CDFG In this step, the given RTL is parsed and each concurrent block of RTL code is transformed into a CDFG data structure. Figure 6 shows the transformation of an "always @()" block of a Verilog code to its corresponding CDFG. Next, small CDFGs are merged (whenever possible) to build larger combined CDFGs. For example, all CDFGs corresponding to non-blocking assignments to clocked registers can be combined together without any change of the functionality. This procedure creates larger CDFGs with substantially more number of nodes than the constituent CDFGs, which helps to obfuscate the *hosted* mode-control FSM better.

"Hosting" the Mode Control FSM Instead of having a stand-alone mode control FSM as in [9], the state elements of the mode-control FSM can be hosted in existing registers in the design to increase the level of obfuscation. This way, the FSM becomes an integral part of the design, instead of controlling the circuit as a structurally isolated element. An example is shown in Fig. 7, where the 8-bit register *reg1*, referred as the "host register", has been expanded to 12-bits to host the mode-control FSM in its left 4-bits. When these 4-bits are set at values $4'h1$ or $4'h2$, the circuit is in its *normal* mode, while the circuit is in its *obfuscated mode*



```
always @(posedge clk or negedge rst)
begin
   if (!rst) reg1 <= 8'b0;
   else begin
    if (reg1==8'h12 & in1==4'h1)
          reg1 <= 8'h1a;
    else if (reg1==8'hcc & in1==4'h2)
          reg1 <= 8'hbb;
    else if (reg1==8'h2f & in2==4'ha)
          reg1 <= 8'hde;
     else reg1 <= 8'hef;
   end
end
```

```
always @(posedge clk or negedge rst) begin
   if (!rst) reg1 <= 12'b0;
   else begin
    if ((reg1==12'h212 | reg1==12'h112)
        & in1==4'h1) reg1 <= 12'h11a;
    else if ((reg1==12'ha0f | reg1==12'hb0f)
        & in1==4'h2) reg1 <= 12'hb11;
    else if ((reg1==12'h1cc | reg1==12'h2cc)
        & in1==4'h2) reg1 <= 12'h2bb;
     else if ((reg1==12'h22f | reg1==12'h122f)
        & in2==4'ha) reg1 <= 2'h2de;
     else if ((reg1==12'hb22 | reg1==12'ha22)
         & in2==4'hf) reg1 <= 12'hb66;
     else if (^in1) reg1 <= 12'h1ef;
     else reg1 <= 12'h2ef;
   end
end
```

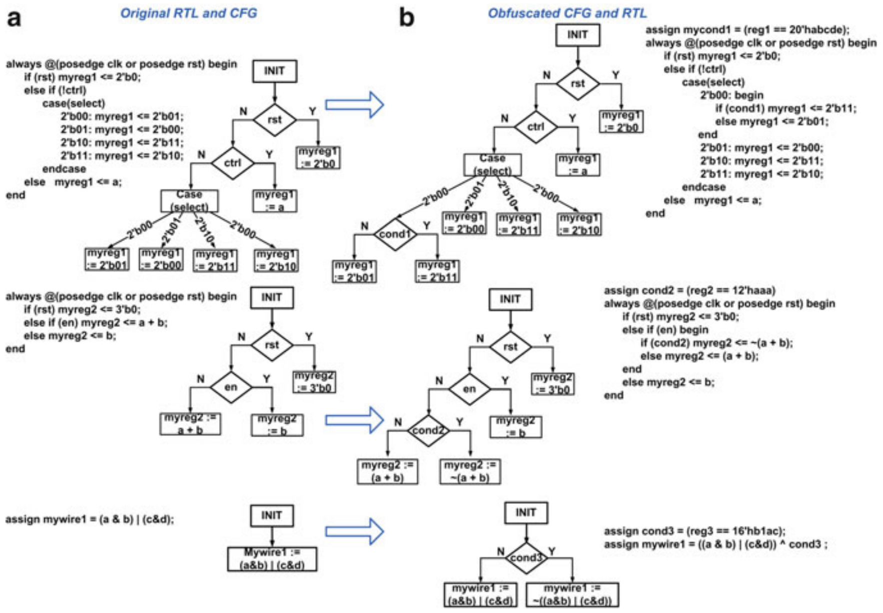
Fig. 7 Example of hosting the registers of the mode-control FSM [12]

when they are at $4'ha$ or $4'hb$. Note that extra RTL statements have been added to make the circuit functionally equivalent in the *normal mode*. The obfuscation level is improved by distributing the mode-control FSM state elements in a non-contiguous manner inside one or more registers, if possible.

Modifying CDFG Branches After the FSM has been hosted in a set of selected *host registers*, several CDFG nodes are modified using the control signals generated from this FSM. The nodes with large fanout cones are preferentially selected for modification, since this ensures maximum change in functional behavior at minimal design overhead. Three example modifications of the CDFGs and the corresponding RTL statements are shown in Fig. 8. The registers *reg1*, *reg2* and *reg3* are the *host registers*. Three "case()", "if()" and "assign" statements in Fig. 8a are modified by the mode-control signals *cond1*, *cond2* and *cond3*, respectively. These signals evaluate to logic-1 only in the *obfuscation mode* because the conditions *reg1*$=20'habcde$, *reg2*$=12'haaa$ and *reg3*$=16'hb1ac$ correspond to states which are only reachable in the *obfuscation mode*. Figure 8b shows the modified CDFGs and the corresponding CDFG statements.

Besides changing the control-flow of the circuit, functionality is also modified by introducing additional datapath components. However, such changes are done in a manner that ensures sharing of the additional resources during synthesis. This is important since datapath components usually incur large hardware overhead. An example is shown in Fig. 9, where the signal *out* originally computes $(a + b) \times (a - b)$. However, after modification of the RTL, it computes $(a + b)$ in the *obfuscated mode*, allowing the adder to be shared in the two modes, and the outputs of the multiplier and the adder to be multiplexed.

Generating Obfuscated RTL After the modifications have been preformed on the CDFG, the obfuscated RTL is generated from the modified CDFGs, by traversing each of them in a *depth-first* manner. Figure 10a shows an example RTL code



```
always @(posedge clk or posedge rst) begin
    if (rst) myreg1 <= 2'b0;
    else if (!ctrl)
        case(select)
            2'b00: myreg1 <= 2'b01;
            2'b01: myreg1 <= 2'b00;
            2'b10: myreg1 <= 2'b11;
            2'b11: myreg1 <= 2'b10;
        endcase
    else   myreg1 <= a;
end
```

```
assign mycond1 = (reg1 == 20'habcde);
always @(posedge clk or posedge rst) begin
    if (rst) myreg1 <= 2'b0;
    else if (!ctrl)
        case(select)
            2'b00: begin
                if (cond1) myreg1 <= 2'b11;
                else myreg1 <= 2'b01;
            end
            2'b01: myreg1 <= 2'b00;
            2'b10: myreg1 <= 2'b11;
            2'b11: myreg1 <= 2'b10;
        endcase
    else   myreg1 <= a;
end
```

```
always @(posedge clk or posedge rst) begin
    if (rst) myreg2 <= 3'b0;
    else if (en) myreg2 <= a + b;
    else myreg2 <= b;
end
```

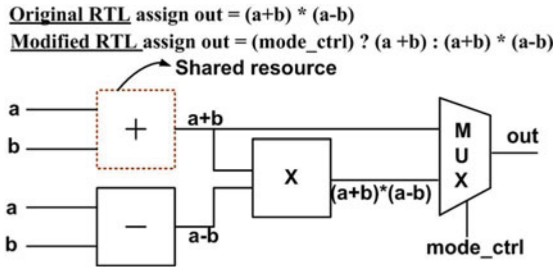
```
assign cond2 = (reg2 == 12'haaa)
always @(posedge clk or posedge rst) begin
    if (rst) myreg2 <= 3'b0;
    else if (en) begin
        if (cond2) myreg2 <= ~(a + b);
        else myreg2 <= (a + b);
    end
    else myreg2 <= b;
end
```

```
assign mywire1 = (a & b) | (c&d);
```

```
assign cond3 = (reg3 == 16'hb1ac);
assign mywire1 = ((a & b) | (c&d)) ^ cond3 ;
```

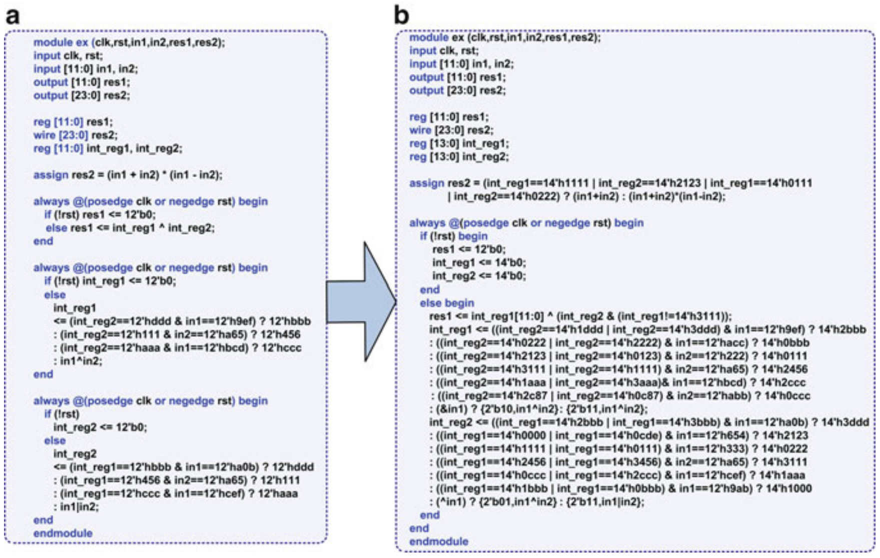
Fig. 8 Examples of control-flow obfuscation: (**a**) original RTL, CDFG; (**b**) obfuscated RTL, CDFG [12]

Fig. 9 Example of datapath obfuscation allowing resource sharing [12]



Original RTL assign out = (a+b) * (a-b)
Modified RTL assign out = (mode_ctrl) ? (a +b) : (a+b) * (a-b)

and Fig. 10b shows its corresponding obfuscated versions. A 4-bit FSM has been *hosted* in registers *int_reg1* and *int_reg2*. The conditions *int_reg1*[13:12]= $2'b00$, *int_reg1*[13:12]= $2'b01$, *int_reg2*[13:12]= $2'b00$ and *int_reg1*[13:12]= $2'b10$ occur only in the obfuscated mode. The initialization sequence is *in1*= $12'h654$ → *in2*= $12'h222$ → *in1*= $12'h333$ → *in2*= $12'hacc$ → *in1*= $12'h9ab$. Note the presence of *dummy state transitions* and out-of-order state transition RTL statements. The outputs *res1* and *res2* have been modified by two different *modification signals*. Instead of allowing the inputs to appear directly in the sensitivity list of the "if()" statements, it is possible to derive internal signals (similar to the ones shown in Fig. 8b) with complex Boolean expressions which are used to perform the modifications. The output *res1* has been modified following the datapath modification approach using resource sharing.


a

```
module ex (clk,rst,in1,in2,res1,res2);
input clk, rst;
input [11:0] in1, in2;
output [11:0] res1;
output [23:0] res2;

reg [11:0] res1;
wire [23:0] res2;
reg [11:0] int_reg1, int_reg2;

assign res2 = (in1 + in2) * (in1 - in2);

always @(posedge clk or negedge rst) begin
  if (!rst) res1 <= 12'b0;
   else res1 <= int_reg1 ^ int_reg2;
end

always @(posedge clk or negedge rst) begin
  if (!rst) int_reg1 <= 12'b0;
  else
    int_reg1
    <= (int_reg2==12'hddd & in1==12'h9ef) ? 12'hbbb
    : (int_reg2==12'h111 & in2==12'ha65) ? 12'h456
    : (int_reg2==12'haaa & in1==12'hbcd) ? 12'hccc
    : in1^in2;
end

always @(posedge clk or negedge rst) begin
  if (!rst)
    int_reg2 <= 12'b0;
  else
    int_reg2
    <= (int_reg1==12'hbbb & in1==12'ha0b) ? 12'hddd
    : (int_reg1==12'h456 & in2==12'ha65) ? 12'h111
    : (int_reg1==12'hccc & in1==12'hcef) ? 12'haaa
    : in1|in2;
end
endmodule
```

b

```
module ex (clk,rst,in1,in2,res1,res2);
input clk, rst;
input [11:0] in1, in2;
output [11:0] res1;
output [23:0] res2;

reg [11:0] res1;
wire [23:0] res2;
reg [13:0] int_reg1;
reg [13:0] int_reg2;

assign res2 = (int_reg1==14'h1111 | int_reg2==14'h2123 | int_reg1==14'h0111
         | int_reg2==14'h0222) ? (in1+in2) : (in1+in2)*(in1-in2);

always @(posedge clk or negedge rst) begin
  if (!rst) begin
    res1 <= 12'b0;
    int_reg1 <= 14'b0;
    int_reg2 <= 14'b0;
  end
  else begin
   res1 <= int_reg1[11:0] ^ (int_reg2 & (int_reg1!=14'h3111));
   int_reg1 <= ((int_reg2==14'h1ddd | int_reg2==14'h3ddd) & in1==12'h9ef) ? 14'h2bbb
   : ((int_reg2==14'h0222 | int_reg2==14'h2222) & in1==12'hacc) ? 14'h0bbb
   : ((int_reg2==14'h2123 | int_reg2==14'h0123) & in2==12'h222) ? 14'h0111
   : ((int_reg2==14'h3111 | int_reg2==14'h1111) & in2==12'ha65) ? 14'h2456
   : ((int_reg2==14'h1aaa | int_reg2==14'h3aaa)& in1==12'hbcd) ? 14'h2ccc
    : ((int_reg2==14'h2c87 | int_reg2==14'h0c87) & in2==12'habb) ? 14'h0ccc
   : (&in1) ? {2'b10,in1^in2}: {2'b11,in1^in2};
   int_reg2 <= ((int_reg1==14'h2bbb | int_reg1==14'h3bbb) & in1==12'ha0b) ? 14'h3ddd
   : ((int_reg1==14'h0000 | int_reg1==14'h0cde) & in1==12'h654) ? 14'h2123
   : ((int_reg1==14'h1111 | int_reg1==14'h0111) & in1==12'h333) ? 14'h0222
   : ((int_reg1==14'h2456 | int_reg1==14'h3456) & in2==12'ha65) ? 14'h3111
   : ((int_reg1==14'h0ccc | int_reg1==14'h2ccc) & in1==12'hcef) ? 14'h1aaa
   : ((int_reg1==14'h1bbb | int_reg1==14'h0bbb) & in1==12'h9ab) ? 14'h1000
   : (^in1) ? {2'b01,in1^in2} : {2'b11,in1|in2};
  end
end
endmodule
```

Fig. 10 Example of RTL obfuscation by CDFG modification: (**a**) original RTL; (**b**) obfuscated RTL [12]

3.2.2 Embedding Authentication Features

Authentication features might be embedded in the RTL by the same principle as described in Sect. 3.1.2. RTL statements describing the state transitions of the *authentication FSM* can be integrated with the existing RTL in the same way the statements corresponding to the *obfuscation FSM* is hidden. In case the *unused states* are difficult to derive from the original RTL, it can be synthesized to a gate-level netlist and the same technique based on *sequential justification* as described in Sect. 3.1.2 might be applied.

3.3 Comparison Between the Two Approaches

Table 1 compares the relative advantages and disadvantages of the two proposed techniques. Although the de-compilation based approach potentially can hide the modifications better than the direct RTL modification based approach (as shown by our theoretical analysis of their obfuscation levels in Sect. 4 and by our simulation results), it also loses major RTL constructs and creates a description of the circuit which might result in an unoptimized implementation on re-synthesis. Hence, we provide the IP designer with a choice where either of the techniques might be chosen based on the designer's priority. For example, if the IP is going to released to a untrustworthy SoC design houses with a prior record of practicing IP piracy, the

Table 1 Comparison of de-compilation based and CDFG based obfuscation approaches

Approach	Advantages	Disadvantages
STG modification	(a) Higher level of obfuscation	(a) Loses major RTL constructs (b) Greater hardware and computational overheads
CDFG modification	(a) Works directly on RTL descriptions (b) Preserves RTL constructs	(a) Hiding modifications is more challenging

STG modification system might be used. On the other hand if the IP is to released to a comparatively more trustable SoC design house where the design specifications are very aggressive, the CDFG modification based approach might be used.

3.4 Obfuscation-Based Secure SoC Design Flow

The proposed obfuscation based techniques can be utilized to develop a piracy-proof SoC design and manufacturing flow, as described in [9]. The SoC designer receives different obfuscated IPs from the same or different vendors, and then integrates them on the SoC. A special integrated controller module receives patterns from the primary inputs and controls the systematic initialization of the IP modules in the SoC. The system designer integrates several such SoCs on a board, and uses initialization sequences from a secure, tamper-proof microcontroller [20] enhanced with a secure EEPROM [19] to enable all the SoCs. This prevents unauthorized reading/observations of the initialization sequence keys during the initialization phase. Typically, the latency incurred in the initialization can be easily masked in the latency inherent in a ”bootup” or a similar process. Thus, the end user remains oblivious to the embedded security measures in the SoCs. By supporting obfuscated IP cores in the design flow, all the parties (the IP vendor, the SoC designer and the system designer) are benefitted by being protected from piracy.

4 Measure of Obfuscation Level

In general, it is a difficult problem to detect through analysis the complete functionality of a given circuit, irrespective of the form (RTL/gate-level) it has been described in. A related problem of detecting whether a given circuit has a *hardware Trojan* inserted in it has been shown capable of evading most traditional testing and verification techniques, since such analysis is equivalent to the *Halting Problem*, according to *Rice’s Theorem* [24, 37]. Next, we present a quantitative analysis on the level of difficulty for breaking the proposed obfuscation-based protection.

We try to analyze the security of the proposed schemes against three different attacks: (a) attacks based on manual effort of visually trying to identify the modifications in the RTL; (b) attacks through simulation-based functional analysis, and (c) attacks through structural analysis and reverse-engineering. In all the cases, we assume a challenging situation for the IP owner where the adversary has access to an unobfuscated version of the original RTL.

4.1 Manual Attacks by Visual Inspection

To estimate the obfuscation level against a manual mode of attack, we propose a new metric called *semantic obfuscation metric* (M_{sem}), which depicts how many of the original high level RTL constructs have been replaced by new ones. We define M_{sem} by:

$$M_{sem} = \frac{abs(N_{c,orig} + N_{w,orig} + N_{e,obfus} - N_{raw,obfus})}{max(\{N_{c,orig} + N_{w,orig} + N_{e,obfus}\}, N_{raw,obfus})} \tag{2}$$

where $N_{c,orig}$ is the total number of high-level RTL constructs in the original RTL; $N_{e,obfus}$ is the number of extra state elements included in the obfuscated design; $N_{w,orig}$ is the total number of internal `wire` declarations in the original RTL and $N_{raw,obfus}$ is the number of `reg`, `assign` and `wire` declarations in the obfuscated RTL. Note that $0 \leq M_{sem} \leq 1$, with a higher value implying better obfuscation. M_{sem} represents a measure of semantic difference between the obfuscated and the unobfuscated versions of the RTL, by taking into consideration the constructs introduced in the obfuscated code and the constructs removed from the original code. This is the weakest attack, with the adversary having very little chance of figuring out the obfuscation scheme for large RTLs which have undergone a complete change of the "look-and-feel".

4.2 Simulation-Based Attacks

For a logic simulation based approach where random input vectors are sequentially applied to take the circuit to the *normal mode*, the probability of discovering the initialization key sequence is $\frac{1}{2^{M \cdot N}}$ for a circuit with M primary input ports and a length-N initialization key sequence. For example, in a circuit with $M = 64$ primary inputs and a length $N = 4$ initialization key sequence, this probability is $\sim 10^{-77}$. In practice, most IPs will have larger number of primary inputs and the length N can be made larger, resulting in smaller detection probability. Thus, we can claim that it is extremely challenging to break the scheme using simulation based reverse-engineering.

4.3 Structural Analysis Based Attack

For the structural analysis based attack, the two proposed obfuscation schemes present different challenges to an adversary. For the STG modification scheme, the adversary has to analyze the circuit in terms of the Boolean logic structure of the internal nodes, while in the CDFG modification based scheme, the adversary has to analyze the high-level RTL structure of the code. This is the strongest attack, and to be acceptable, the proposed obfuscation approaches must provide adequate protection against this attack. We describe the complexity of the two analyses below.

4.3.1 Structural Analysis Against STG Modification

Analysis based on structure of the internal nodes is most conveniently done by the construction and manipulation of Reduced Ordered Binary Decision Diagrams (ROBDD) [6] corresponding to the internal circuit nodes. To detect the node modification scheme, the adversary's algorithm must be able to solve several sub-problems in succession. We estimate the computational complexity of each of these sub-problems below to derive an estimate of the computational complexity of the entire problem.

Let the total number of primary outputs of the circuit be P, the total number of state elements in the original circuit be S and the total number of state elements inserted in the modified circuit be T. Then, it is sufficient to analyze the structures of these $(P + S + T)$ nodes between the original and the modified designs, out of which $(P + S)$ are also present in the original design. Suppose, the adversary finds F nodes out of these $(P + S + T)$ nodes to have contrasting logic structures by a ROBDD-based analysis. This dissimilarity is due to two reasons: (a) direct effect of the node modification scheme described in Sect. 3 on some of these nodes, and (b) indirect effect of these modified nodes on other nodes. Either way, from Eq. (1), the affected nodes would have their values inverted only if simultaneously $en = 1$ and $g = 1$. To isolate the inserted FSM, the adversary must detect this node modification scheme for each dissimilar node.

Finding the Correct ROBDD Representation of the Modified Nodes To detect the effect of a particular *en* signal originating from the inserted state machine on a modified node, the adversary's algorithm should be able to represent the ROBDD of the modified node with the *en* signal as the root node, as shown in Fig. 11. Improving the variable ordering to minimize the size of a BDD is an *NP-complete* problem [5]. Hence, it follows that the computational complexity to find a particular representation of the ROBDD of the modified function which has *en* as the root node is also *NP-complete* with respect to the number of variables in the Boolean logic expression for the node. Hence, deciphering the modification scheme for a modified node with fanin cone size f_i has a computational complexity $O(2^{f_i})$.

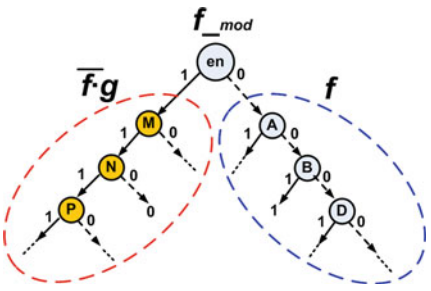


Fig. 11 Binary decision diagram (BDD) of a modified node

Graph Isomorphism Comparison After the ROBDD of the modified node has been expressed in the form shown in Fig. 11, each sub-graph below the node *en* should be compared with the ROBDD graph for f for isomorphism. Proving graph isomorphism is a problem with computational complexity between P and NP, with the best-known heuristic having complexity $2^{O(\sqrt{n \log n})}$ for a graph with n vertices [27]. Hence, establishing the equivalence for f through graph isomorphism has a computational complexity $2^{O(\sqrt{f_i \log f_i})}$ for a node with fanin cone size f_i. Let $\overline{f_i}$ be the average fanin cone size of the failing verification nodes. Hence, overall this sub-problem has a computational complexity $O(2^{\overline{f_i}} \cdot 2^{O(\sqrt{\overline{f_i} \log \overline{f_i}})})$, which must be solved for each of the F dissimilar nodes.

Compare Point Matching So far we have assumed that the adversary would be able to associate the dissimilar nodes in the obfuscated design with the corresponding nodes in the original design and would then try to decipher the obfuscation scheme. This is expected to be relatively easy for the primary output ports of the IP because the IP vendor must maintain a standard interface even for the obfuscated version of the IP, and hence the adversary can take advantage of this name matching. However, the names of the state elements can be changed arbitrarily by the IP vendor and hence, finding the corresponding state elements to compare is a *functional compare point matching* problem [2], which is extremely computationally challenging because of the requirement to search through $(S_N)!$ combinations, where S_N is the number of dissimilar state elements. Hence, we propose the following metric to quantify the level of protection of the proposed STG modification based obfuscation scheme in providing protection against structural analysis:

$$M_{str} = F \cdot 2^{\overline{f_i}} \cdot 2^{\sqrt{\overline{f_i} \log \overline{f_i}}} + (S_N)! \tag{3}$$

Observations from the Metric From the above metric, the following can be observed which act as a guide to the IP designer to design a well-obfuscated hardware IP following the STG modification based scheme:

1. Those nodes which have larger fanin cones should be preferably modified because this would increase $\overline{f_i}$ in Eq. (3), thus increasing M_{str}.

2. An inserted FSM with larger number of flip-flops increases its obfuscation level because S_N increases. Also, as shown previously in this section, there is an exponential dependance of the probability of breaking the scheme by simulation based reverse-engineering on the length of the initialization key sequence. Hence, it is evident that FSM design and insertion to attain high levels of obfuscation incur greater design overhead. Thus the IP designer must trade-off between design overhead and the level of security achievable through obfuscation.
3. Modification of a larger number of nodes increases F, which in turn increases the level of obfuscation.

4.3.2 Structural Analysis Against CDFG Modification Based Approach

The structural analysis of the CDFG modification based obfuscation scheme is estimated by the degree of difficulty faced by an adversary in discovering the *hosted* mode-control FSM and the modification signals. Consider a case where n mode-control FSM state-transition statements have been hosted in a RTL with N blocking/non-blocking assignment statements. However, the adversary does not know a-priori how many registers host the mode-control FSM. Then, the adversary must correctly figure out the hosted FSM state transition statements from one out of $\sum_{k=1}^{n}\binom{N}{k}$ possibilities. Again, each of these choices for a given value of k has $k!$ associated ways to arrange the state transitions (so that the *initialization key sequence* is applied in the correct order). Hence, the adversary must correctly identify one out of $\sum_{k=1}^{n}\left(\binom{N}{k}\cdot k!\right)$ possibilities. The other feature that needs to be deciphered to break the scheme are the mode control signals. Let M be the total number of blocking, non-blocking and dataflow assignments in the RTL, and let m be the size of the modification signal pool. Then, the adversary must correctly choose m signals out of M, which is one out of $\binom{M}{m}$ choices. Combining these two security features, we propose the following metric to estimate the complexity of the structural analysis problem for the CDFG modification based design:

$$M_{str} = \sum_{k=1}^{n}\left(\binom{N}{k}\cdot k!\right)\cdot\binom{M}{m} \tag{4}$$

A higher value of M_{str} indicates a greater obfuscation efficiency. As an example, consider a RTL with values $N = 30$, $M = 100$, in which a FSM with parameter $n = 3$ is hosted, and let $m = 20$. Then, $M_{obf} \approx 1.35\times10^{25}$. In other words, the probability of the hacker reverse-engineering the complete scheme is about 1 in 10^{25}. In practice, the values of n and M would be much higher in most cases, making M_{str} larger and thus tougher for the hacker to reverse-engineer the obfuscation scheme.

4.4 Quantitative Comparison

The value of M_{str} is expected to be better in the STG modification based approach, because of the exponential dependance of this metric on the average fanin cone size of the modified nodes [Eq. (3)], compared to the combinatorial dependance on the number of RTL statements for the CDFG modification based approach [Eq. (4)]. The value of M_{sem} is again expected to be superior in the STG modification based approach because the changes in the appearance of the RTL is more drastic in this case, whereas the CDFG modification based approach makes comparatively lesser changes to the high-level RTL constructs, as it is based on intelligent utilization of mostly existing RTL constructs. However, the run-time of the obfuscation algorithm is expected to be higher in the STG modification based approach, because it performs recursive backtracking to construct the logic equations of the internal nodes. All these predicted trends were supported by our simulation results presented in Sect. 5.

5 Results

In this section we first describe the automated design flows for the two proposed obfuscation techniques, followed by the simulation results of application of the techniques to open-source IPs and benchmark circuits.

5.1 Design Flow Automation

Figure 12 shows the entire STG modification based RTL obfuscation design flow. The design flow starts with the *compilation* of RTL description of the IP core to a unmapped, unoptimized gate-level Verilog netlist. The maximum allowable area overhead is entered as a design constraint, from which the maximum number of modifiable nodes (N_{max}) is estimated. Additionally, the tool has a list of user-mentioned constructs and macros in a forward annotation file. These elements are preserved during the RTL compilation and de-compilation processes by treating them as *don't touch* modules. The N_{max} nodes to be modified are chosen based on the algorithm described in [9], which ensures maximum perturbation of the design. The modified netlist is re-synthesized and the resultant netlist is then de-compiled to a RTL code. The names of the internal nodes and instances are changed by a simple string substitution scheme.

Figure 13 shows the steps of the proposed CDFG modification based design obfuscation methodology. The input to the flow is the original RTL, the desired obfuscation level represented by the obfuscation metrics (M_{sem} and M_{str}), and the maximum allowable area overhead. It starts with the design of the mode-control

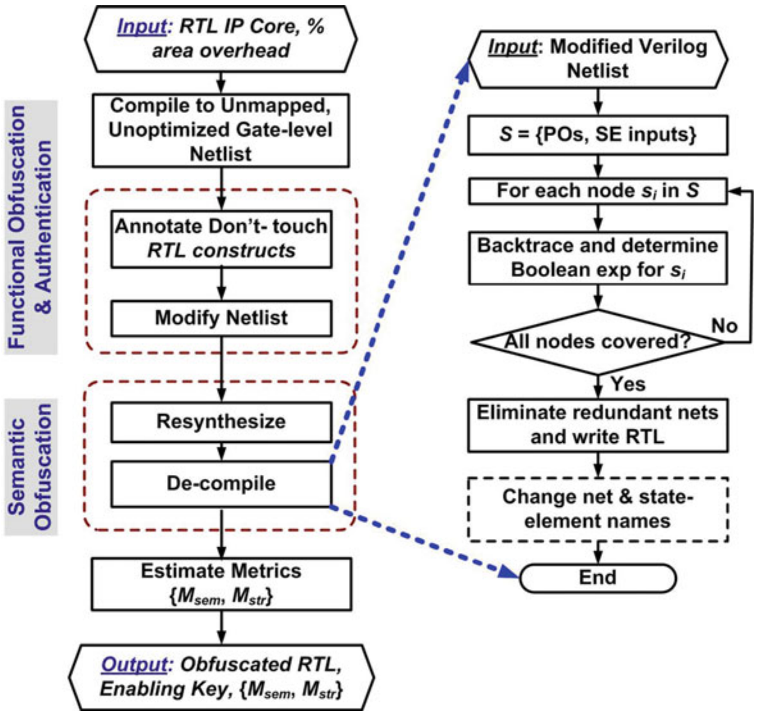

Input: RTL IP Core, % area overhead
Compile to Unmapped, Unoptimized Gate-level Netlist
Annotate Don't- touch RTL constructs
Modify Netlist
Resynthesize
De-compile
Estimate Metrics {M_{sem}, M_{str}}
Output: Obfuscated RTL, Enabling Key, {M_{sem}, M_{str}}
Functional Obfuscation & Authentication
Semantic Obfuscation
Input: Modified Verilog Netlist
S = {POs, SE inputs}
For each node s_i in S
Backtrace and determine Boolean exp for s_i
All nodes covered?
No
Yes
Eliminate redundant nets and write RTL
Change net & state-element names
End

Fig. 12 Flow diagram for the proposed STG modification based RTL obfuscation methodology [11]

FSM based on the target M_{sem} and M_{str}. The output of this step are the specifications of the FSM which include its state transition graph, the state encoding, the pool of modification signals, and the *initialization key sequence*. Random state encoding and a random *initialization key sequence* are generated to increase the security. Note that in the STG modification based approach, we do not start with explicit target values of the metrics, because these two parameters cannot be predicted a-priori in this technique. However, the target area overhead is an indirect estimate of these parameters, and the de-compilation process automatically ensures a high value of M_{sem}, while an optimal node modification algorithm ensures a high value of M_{str}.

5.2 Simulation Results

5.2.1 Simulation Setup

The above design flows were developed using C and the TCL scripting language and was directly integrated in the *Synopsys Design Compiler* environment. Synthesis

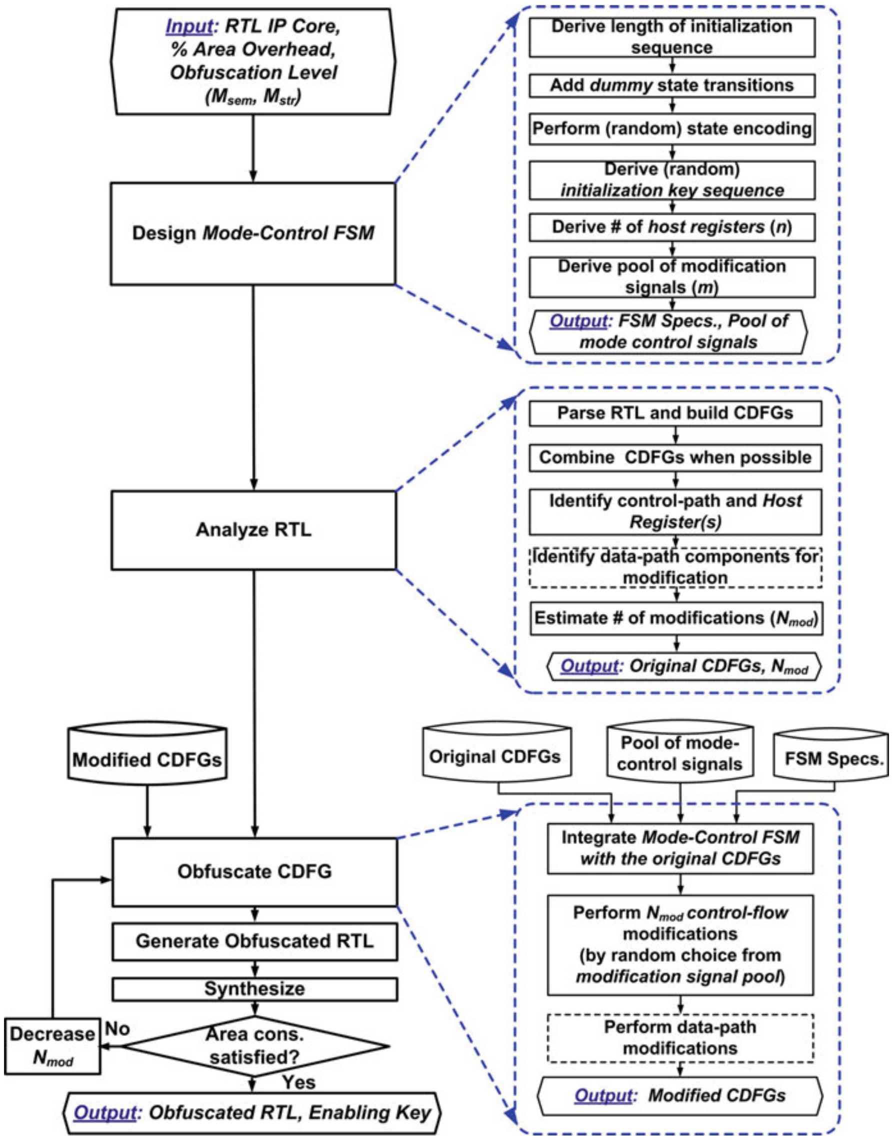

Input: RTL IP Core, % Area Overhead, Obfuscation Level (M_{sem}, M_{str})
Design Mode-Control FSM
Derive length of initialization sequence
Add dummy state transitions
Perform (random) state encoding
Derive (random) initialization key sequence
Derive # of host registers (n)
Derive pool of modification signals (m)
Output: FSM Specs., Pool of mode control signals
Analyze RTL
Parse RTL and build CDFGs
Combine CDFGs when possible
Identify control-path and Host Register(s)
Identify data-path components for modification
Estimate # of modifications (N_{mod})
Output: Original CDFGs, N_{mod}
Modified CDFGs
Original CDFGs
Pool of mode-control signals
FSM Specs.
Obfuscate CDFG
Integrate Mode-Control FSM with the original CDFGs
Perform N_{mod} control-flow modifications (by random choice from modification signal pool)
Perform data-path modifications
Output: Modified CDFGs
Generate Obfuscated RTL
Synthesize
Area cons. satisfied?
No
Yes
Decrease N_{mod}
Output: Obfuscated RTL, Enabling Key

Fig. 13 Flow diagram for the proposed CDFG modification based RTL obfuscation methodology [12]

was performed using *Synopsys Design Compiler*, using a LEDA 250 nm standard-cell library. All formal equivalence checking was performed using *Synopsys Formality*. All work was performed on a Linux workstation with 2 GB of main memory and a dual-core 1.5 GHz processor.

A FSM with a length-4 initialization key sequence was designed for mode-control in both the schemes. In the STG modification based scheme, we chose high fan-in internal nodes as MKFs, as described in [9]. The STG modification scheme was applied on three open-source Verilog IP cores, viz. “Data Encryption Standard” (DES), “Advanced Encryption Standard” (AES) and “Discrete Cosine Transform” (FDCT) collected from [34]. The CDFG modification based obfuscation technique was applied for two Verilog IP cores—a single precision IEEE-754 compliant floating-point unit (“FPU”), and a 12-bit RISC CPU (“TCPU”), both obtained from [34]. For the STG modification based approach, each gate-level modified design was verified using *Formality* with the corresponding de-compiled RTL to verify the correctness of the procedure.

5.2.2 Obfuscation Efficiency and Overheads

Table 2 shows the structural and semantic obfuscation metrics and design overheads for 10 % target area overhead constraint for each module. Note that the value of the M_{str} metric is in a logarithmic scale. The effectiveness of the schemes was very high; most of the modified designs considered reported close to 100 % equivalence checking failure (for the primary outputs and outputs of state elements) when compared with their unmodified counterparts, thus having a very high value of the computational complexity metric M_{str}. However, the value of M_{str} is orders of magnitude higher for the STG modification based approach, as predicted in Sect. 4. The value of M_{sem} was very close to the ideal value of 1.0 for the STG modification based approach; however, it is closer to 0.75 on average for the CDFG modification based approach. Again, this is an expected trend as predicted in Sect. 4.

For all the individual circuit modules, the observed area overhead was less than 10 %, the power and delay overheads were within acceptable limits (target delay overhead was set at 0 % for the CDFG modification based scheme). The maximum run-times of the obfuscation programs for the individual modules was 29 s for the CDFG modification based approach, and 37 s for the STG modification based approach. Table 3 shows the overall design overheads after re-synthesis of the multi-module IP cores from the obfuscated RTL, which are again all within acceptable limits.

5.3 *Effect of Key Length*

The security offered by the two proposed approaches increases with the increase in the key length. We investigate the following aspects of the proposed techniques (a) design and performance overheads to support multiple-length keys, and (b) effect of increasing key lengths on design and performance overheads. We compare the proposed techniques with hardware implementations of the AES encryption/decryption algorithm.

Table 2 Functional and semantic obfuscation efficiency and overheads for IP cores for 10 % area overhead target (CDFG modification based results at iso-delay)

STG modification based approach									
			Obfuscation efficiency			Design overhead			
IP cores	Sub-modules	Nodes modified (%)	Failing verif. nodes (%)	M_{sem}	$\log_{10} M_{str}$	Area (%)	Delay (%)	Power (%)	Run time (s)
DES	key_sel	0.93	100.00	0.98	51.56	6.65	1.39	4.85	32
	crp	0.83	100.00	0.91	55.87	5.54	0.66	5.43	37
AES	Key expand	0.95	90.30	0.92	43.17	5.29	0.00	4.56	28
	Sbox	0.95	100.00	0.96	45.78	4.95	2.42	5.31	29
	Inverse Sbox	0.97	85.25	0.94	46.22	5.51	2.60	5.62	35
FDCT	DCT	0.90	88.95	0.96	60.19	4.64	1.00	5.06	27
	ZIGZAG	0.95	100.00	0.92	52.12	5.74	0.88	5.67	30
CDFG modification based approach									
			Obfuscation efficiency			Design overhead			
IP cores	Sub-modules	# of modifications	Failing verif. nodes (%)	M_{sem}	$\log_{10} M_{str}$	Area (%)	Delay (%)	Power (%)	Run time (s)
FPU	post_norm	20	98.16	0.69	36.81	8.22	0.00	9.14	27
	pre_norm	20	94.16	0.70	32.91	9.39	0.00	9.79	25
	pre_norm_fmul	20	90.00	0.77	23.13	8.30	0.00	9.69	20
	except	10	100.00	0.73	23.16	7.56	0.00	8.73	14
TCPU	control_wopc	20	92.79	0.75	42.12	8.74	0.00	8.97	29
	mem	10	97.62	0.71	19.69	8.29	0.00	9.76	15
	alu	10	97.62	0.81	15.01	9.59	0.00	9.88	15

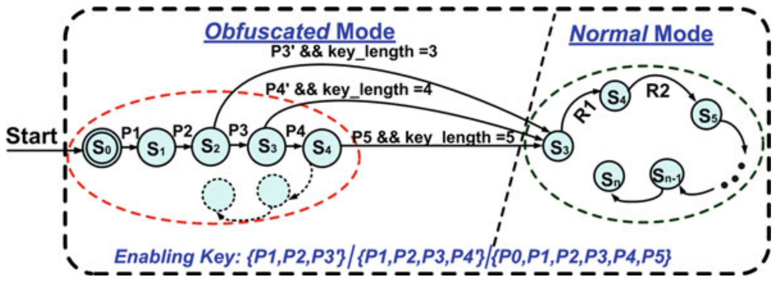


Fig. 14 Scheme with *initialization key sequences* of varying length (3, 4 or 5)

Table 3 Overall design overheads for obfuscated IP cores (STG modification based results at iso-delay)

IP core	Area (%)	Delay (%)	Power (%)
STG modification based approach			
DES	6.09	1.10	5.05
AES	5.25	2.00	5.45
FDCT	5.22	0.95	5.55
Average	5.52	1.35	5.35
CDFG modification based approach			
FPU	8.48	9.36	0.00
TCPU	8.54	9.73	0.00
Average	8.51	9.55	0.00

Table 4 Area overhead for multi-length key support

Scheme	Area overhead (%)
STG modification	2.26 (av.)
CDFG modification	3.16 (av.)
Ref. [21] (AES 48-cycle core)	17.88
Ref. [21] (AES 96-cycle core)	24.59

5.3.1 Support for Multiple-Length Keys

Most commercially available IP cores for AES can be operated in three different modes with three different input key lengths—128 bit, 192 bit and 256 bit [1, 21]. This flexibility allows the SoC designers to trade-off between the available security (which increases with increase of key length) and performance (which decreases with increase of key length). Usually, a "*key_length*" input control signal determines the input key length. The same feature can be implemented in our proposed techniques, where the length of the initialization key sequence can be varied. Figure 14 shows such a system which supports initialization key sequences of length 3, 4 or 5. However, in case of commercially available AES cores, this flexibility comes at a price—the multi-key IP core versions usually have greater area than the baseline designs supporting only a single key length [21].

Table 4 shows the area overhead effect of supporting multiple keys lengths on the proposed schemes for the IP modules presented in Table 3, compared to

Table 5 Comparison of throughput with increasing key length

	Decrease in throughput (%)	
Scheme	1.5X key length	2X key length
STG modification	2.70 (av.)	3.96 (av.)
CDFG modification	3.07 (av.)	4.72 (av.)
Ref. [21] (AES 48-cycle core)	14.10	24.83
Ref. [21] (AES 96-cycle core)	14.04	24.56

two versions of a commercially available AES core [21]. The key lengths for our proposed schemes were 4 (baseline), 6 (1.5X) and 8 (2X), while those for the AES implementations were 128 (baseline), 192 (1.5X) and 256 (2X). From this table, it is clearly evident that the proposed approaches are more scalable than the AES hardware implementations with respect to the increase in key length.

5.3.2 Effect of Increasing Key Length

For symmetric key cryptographic algorithms such as AES, in general the security increases with the length of the key, as the complexity of breaking the encryption is an exponential function of the key length. However, an increasing key length usually results in lower throughput. Similar trends are expected for the two proposed obfuscation schemes with respect to the length of the initialization key sequence. We investigated the scalability of the two proposed techniques with respect to the increase in the length of the initialization key sequence for the proposed approaches vis-a-vis that for commercially available hardware implementations of AES with respect to the key length. Again, for the proposed obfuscation schemes we considered a key length of 4 to the baseline case, while a 128 bit key for AES was considered baseline. Table 5 shows the decrease in throughput with the increase of key length. Once again, the simulation results showed that the proposed schemes had superior scalability of throughput than the hardware implementations of AES when the key length is increased.

6 Discussions

In this section, we describe a technique to decrease the hardware overhead by while utilizing the normally “unused states” (states which never arise during normal operations). We also show how the proposed obfuscation techniques can provide protection against *hardware Trojans*.

Table 6 Area and power overhead for ISCAS-89 benchmarks utilizing unused states (at iso-delay)

Circuit	# of gates	% Ar. Ov.1	% Ar. Ov.2	%Pow. Ov.1	%Pow. Ov.2
s1196	370	18.44	24.84	2.45	6.00
s1238	373	16.31	27.15	4.63	10.71
s1423	505	6.11	6.08	2.015	3.07
s1488	431	12.81	16.93	7.24	12.76
s5378	1102	14.45	18.69	8.76	25.78
s9234	5807	6.53	9.04	9.28	13.15
s13207	2488	7.93	8.17	7.54	16.28
s15850	2983	7.67	8.99	5.63	6.15
s35932	7966	1.34	1.955	6.93	9.49
s38417	8822	0.09	0.56	1.81	5.27
s38584	9019	0.85	3.03	5.10	10.77

6.1 Using Unreachable States During Initialization

Referring to Fig. 5, the states in the *initialization FSM* and the *obfuscation FSM* are encoded by using states which are unreachable during normal operations. By doing this, one can ensure that the circuit would not operate in the correct mode prior to its initialization. This can help to eliminate the need to introduce a separate FSM to control the mode of operation, potentially decreasing the hardware overhead. We present simulation results for a method of obfuscation based on finding unused states for a suite of gate-level sequential circuits, and two open-source RTL IP cores.

Table 6 shows the area and power overhead (at iso-delay) in ISCAS-89 circuits following the de-compilation based methodology where normally unused states of the circuit are used to encode the states in the obfuscated mode. The unused states were found using *sequential justification* by *Synopsys Tetramax*. The results were taken without integrating any mode-control FSM, considering 5–6 randomly chosen state-elements in the original circuit, having an initialization state space with 4 states, and (for result set “1”) an authentication state space consisting of 4 states. State encoding with 6 state elements were required in cases where sufficient unused states are not available from 5 state elements. Table 7 shows the corresponding figures for two open-source IP cores.

6.2 Obfuscation for Protection Against Hardware

Hardware Trojans are malicious modifications in integrated circuits that are difficult to detect by traditional post-manufacturing testing, but can cause devastating malfunctions when the IC is deployed in-field [13]. The ICs are most vulnerable to Trojan insertion by an adversary in potentially untrusted off-shore fabrication facilities, by the reverse-engineering and modification of the GDS-II layout

Table 7 Area and power overhead for IP cores utilizing unused states (at iso-delay)

IP	Module	% Ar. Ov.1	% Ar. Ov.2	%Pow. Ov.1	%Pow. Ov.2
AES	sbox	5.95	7.46	15.19	17.67
	inv_sbox	6.99	8.25	8.98	15.27
	key_expand	3.93	5.47	13.80	15.93
	Overall	4.57	6.07	13.03	16.14
DES	key_sel	4.81	8.91	2.66	5.35
	crp	4.75	6.77	1.75	4.79
	Overall	4.77	7.69	2.03	4.97

database. To evade post-manufacturing testing, the adversary usually ensures that the inserted Trojan circuit is activated and exhibits its malicious effect under rare logic conditions at the internal circuit nodes. However, if the circuit functionality is obfuscated by an initialization key based scheme, then identification of rare conditions becomes tough, and there is a large probability of an inserted Trojan either becoming functionally *benign* or more *detectable* [10]. Because the security features based on obfuscation both the proposed approaches propagate to the GDS-II levels of design abstraction, hence, they can provide protection against hardware Trojans. Note that the obfuscation methodology works in exactly the reverse way compared to hardware Trojans—obfuscated circuits provide incorrect functionality initially, and after the application of a correct key sequence, the actual functionality. In contrast, a circuit containing a hardware Trojan initially exhibits normal functionality, but malfunctions when a sequence (or a single) of inputs causes the hidden hardware Trojan to wake up. A notable work that uses logic obfuscation at a circuit-architecture-level to secure a processor has been described in [36].

7 Conclusions

We have presented two variations of RTL hardware IP protection through key-based obfuscation. The two approaches differ in level of protection, design complexity and overhead. The scheme includes additional hard-to-detect authentication features at low design overhead to increase the level of security. The proposed obfuscation approaches provide active defence against IP infringement at different stages of SoC design and fabrication flow, thus protecting the interests of multiple associated parties. The obfuscation steps can be easily automated and integrated in the IP design flow and it does not affect the test/verification of a SoC design for legal users. We have shown that they incur low design and computational overhead and cause minimal impact on end-user experience. The proposed approaches are easily scalable to large IPs (e.g. processor) and in terms of level of security. Further improvement in hardware overhead can be obtained by utilizing normally unused states of the circuit. The obfuscation based design methodology can also be used to provide protection against hardware Trojan attacks in untrusted fabrication facilities. Future work would include extension of the approach to soft IPs at higher level of abstraction and to achieve protection against hardware Trojan attacks.

References

1. AES (Rijndael) IP Cores. http://www.erst.ch/download/aes_standard_cores.pdf (2011)
2. Anastasakis, D., Damiano, R., Ma, H.K.T., Stanion, T.: A practical and efficient method for compare-point matching. In: Proceedings of the 39th Annual Design Automation Conference, DAC'02, pp. 305–310. ACM, New York (2002)
3. Aucsmith, D.: Tamper resistant software: an implementation. In: Proceedings of the First International Workshop on Information Hiding, pp. 317–333. Springer, London (1996)
4. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '01, pp. 1–18. Springer, London (2001)
5. Bollig, B., Wegener, I.: Improving the variable ordering of OBDDS is NP-complete. IEEE Trans. Comput. **45**, 993–1002 (1996)
6. Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. IEEE Trans. Comput. **35**, 677–691 (1986)
7. Brzozowski, M., Yarmolik, V.N.: Obfuscation as intellectual rights protection in VHDL language. In: Proceedings of the International Conference on Computer Information Systems and Industrial Management Applications, pp. 337–340. IEEE Computer Society, Washington (2007)
8. Castillo, E., Meyer-Baese, U., García, A., Parrilla, L., Lloris, A.: IPP@HDL: efficient intellectual property protection scheme for IP cores. IEEE Trans. Very Large Scale Integr. Syst. **15**, 578–591 (2007)
9. Chakraborty, R.S., Bhunia, S.: HARPOON: an obfuscation-based SoC design methodology for hardware protection. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **28**, 1493–1502 (2009)
10. Chakraborty, R.S., Bhunia, S.: Security against hardware Trojan through a novel application of design obfuscation. In: Proceedings of the 2009 International Conference on Computer-Aided Design, ICCAD '09, pp. 113–116. ACM, New York (2009)
11. Chakraborty, R.S., Bhunia, S.: Security through obscurity: an approach for protecting register transfer level hardware IP. In: Proceedings of the 2009 IEEE International Workshop on Hardware-Oriented Security and Trust, HOST '09, pp. 96–99. IEEE Computer Society, Washington (2009)
12. Chakraborty, R.S., Bhunia, S.: RTL hardware ip protection using key-based control and data flow obfuscation. In: Proceedings of the 2010 23rd International Conference on VLSI Design, VLSID '10, pp. 405–410. IEEE Computer Society, Washington (2010)
13. Chakraborty, R., Narasimhan, S., Bhunia, S.: Hardware Trojan: threats and emerging solutions. In: Proceedings of the IEEE International High Level Design Validation and Test Workshop, HLDVT '09, pp. 166–171 (2009)
14. Chang, H., Atallah, M.J.: Protecting software code by guards. In: Revised Papers from the ACM CCS-8 Workshop on Security and Privacy in Digital Rights Management, DRM '01, pp. 160–175. Springer, London (2002)

15. Charbon, E., Torunoglu, I.: Watermarking techniques for electronic circuit design. In: Proceedings of the 1st International Conference on Digital Watermarking, IWDW'02, pp. 147–169. Springer, Berlin/Heidelberg (2003)
16. Chinese firms favoring soft IP over hard cores. http://www.eetasia.com/ART_8800440032_480100_NT_ac94df1c.HTM (2011)
17. Collberg, C.S., Thomborson, C.: Watermarking, tamper-proffing, and obfuscation: tools for software protection. IEEE Trans. Softw. Eng. **28**, 735–746 (2002)
18. Collberg, C., Thomborson, C., Low, D.: Manufacturing cheap, resilient, and stealthy opaque constructs. In: Proceedings of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '98, pp. 184–196. ACM, New York (1998)
19. DS2432 1kb protected 1-wire EEPROM with SHA-1 engine. http://www.maxim-ic.com/datasheet/index.mvp/id/2914 (2012)
20. DS5002FP secure microprocessor chip. http://www.maxim-ic.com/datasheet/index.mvp/id/2949 (2012)
21. Full datasheet AES-CCM core family for Actel FPGA. http://www.actel.com/ipdocs/HelionCore_AES-CCM_8bit_Actel_DS.pdf (2011)
22. Hou, T., Chen, H., Tsai, M.: Three control flow obfuscation methods for Java software. IEE Proc. Softw. **153**(2), 80–86 (2006)
23. Huang, Y.L., Ho, F.S., Tsai, H.Y., Kao, H.M.: A control flow obfuscation method to discourage malicious tampering of software codes. In: Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS '06, pp. 362–362. ACM, New York (2006)
24. Huffmire, T., Irvine, C., Nguyen, T.D., Levin, T., Kastner, R., Sherwood, T.: Handbook of FPGA Design Security. Springer, Dordrecht (2010)
25. Jakubowski, M.H., Saw, C.W., Venkatesan, R.: Tamper-tolerant software: modeling and implementation. In: Proceedings of the 4th International Workshop on Security: Advances in Information and Computer Security, IWSEC '09, pp. 125–139. Springer, Berlin/Heidelberg (2009)
26. Joepgen, H., Krauss, S.: Software by means of the protprog method. Elecktronik **42**, 52–56 (1993)
27. Johnson, D.S.: The NP-completeness column. ACM Trans. Algoritm. **1**, 160–176 (2005)
28. Kahng, A., Lach, J., Mangione-Smith, W., Mantik, S., Markov, I., Potkonjak, M., Tucker, P., Wang, H., Wolfe, G.: Constraint-based watermarking techniques for design IP protection. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **20**(10), 1236–1252 (2001)
29. Lach, J., Mangione-Smith, W.H., Potkonjak, M.: Robust FPGA intellectual property protection through multiple small watermarks. In: Proceedings of the 36th Annual ACM/IEEE Design Automation Conference, DAC '99, pp. 831–836. ACM, New York (1999)
30. Linn, C., Debray, S.: Obfuscation of executable code to improve resistance to static disassembly. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS '03, pp. 290–299. ACM, New York (2003)
31. Methodology for protection and licensing of HDL IP. http://www.us.design-reuse.com/news/?id=12745&print=yes (2011)
32. Obfuscation by code morphing. http://en.wikipedia.org/wiki/Obfuscated_code#Obfuscation_by_code_morphing (2011)
33. Oliveira, A.: Techniques for the creation of digital watermarks in sequential circuit designs. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **20**(9), 1101–1117 (2001)
34. OpenCores.: http://www.opencores.org (2011)
35. Rajendran, J., Pino, Y., Sinanoglu, O., Karri, R.: Logic encryption: a fault analysis perspective. In: Proceedings of the Conference on Design, Automation and Test in Europe, DATE'12, pp. 953–958. ACM, New York (2012)
36. Rajendran, J., Kanuparthi, A., Zahran, M., Addepalli, S., Ormazabal, G., Karri, R.: Securing processors against insider attacks: a circuit-microarchitecture co-design approach. IEEE Des. Test Comput. **30**(2), 35–44 (2013)

37. Rice, H.: Classes of recursively enumerable sets and their decision problems. Trans. Am. Math. Soc. **74**, 358–366 (1953)
38. Roy, J.A., Koushanfar, F., Markov, I.L.: EPIC: ending piracy of integrated circuits. In: Proceedings of the Conference on Design, Automation and Test in Europe, DATE'08, pp. 1069–1074. ACM, New York (2008)
39. Schulman, A.: Examining the Windows AARD detection code. Dr. Dobb's J. **18**(9) (1993) http://fringe.davesource.com/Fringe/NonZen_Companies/Microsoft/Tactics/1993.09.01.Locks_Out_DrDOS.html
40. Thicket^{TM} family of source code obfuscators. http://www.semdesigns.com (2011)
41. Wirthlin, M.J., McMurtrey, B.: IP delivery for FPGAs using applets and JHDL. In: Proceedings of the 39th Annual Design Automation Conference, DAC '02, pp. 2–7. ACM, New York (2002)
42. Zhuang, X., Zhang, T., Lee, H.H.S., Pande, S.: Hardware assisted control flow obfuscation for embedded processors. In: Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, CASES '04, pp. 292–302. ACM, New York (2004)

Towards Building Trusted Systems: Vulnerabilities, Threats, and Mitigation Techniques

Carson Dunbar and Gang Qu

Abstract Current industry design tools and design flow are developed to improve system performance in terms of speed, size, and power consumption. With security and trust becoming more important for system design, how much can we trust the systems built by these tools? In this chapter, we first give motivational examples to show the potential trust vulnerabilities. Then we focus our investigation on sequential system design from the perspective of finite state machine (FSM). We define the notion of trusted FSM both in the most general sense and specifically when state reachability is used as the trust metric. We give a set of necessary and sufficient conditions to build trusted systems in both cases. More specifically, we find that the traditional FSM synthesis procedure will introduce trust vulnerabilities and cannot guarantee any the logic implementation of the FSM to be trusted. Indeed, we show that not only there exist simple and effective ways to attack a sequential system, it is also possible to insert hardware Trojan into the design without introducing any significant design overhead. We then propose a novel practical approach to designing trusted circuits from the FSM specification. We demonstrate both our findings on the security threats and the effectiveness of our proposed method on the MCNC sequential circuit benchmarks.

1 Introduction

As electronic design automation (EDA) and semiconductors continue to evolve rapidly, a company will not have all the expertise and capability to do in-house design and fabricate the system it wants to build. If the system is designed and fabricated by others, how can the company be convinced that the system is trusted, that is, the delivered system does exactly what the company wants, no more and no less. This is known as the trusted integrated circuits (IC) design challenge, particular for military and civilian systems that require security and access control [1–8].

C. Dunbar • G. Qu (✉)
University of Maryland, College Park, MD, USA
e-mail: cdunbar@umd.edu; gangqu@umd.edu

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_9

Table 1 Truth table of a 3-bit encoder

x	y	z	a	b
0	0	1	0	1
0	1	0	1	0
1	0	0	1	1

When the company gives the system's specification to a design house for layout and then gives the layout information to a foundry for manufacture, the company will lose full control of the system's functionality and specification. An adversary can simply add additional circuitry, known as a hardware Trojan horse, to maliciously modify the system. For example, a Trojan horse can: disable or destroy system components, perform incorrect computation, or leak sensitive information. Most of the existing work on trusted IC design focuses on hardware Trojan detection and prevention [9–13].

We explore the vulnerabilities of both combinational and sequential systems produced by today's design methodology. More specifically, we consider the following questions:

1. When a combinational or sequential system is designed and implemented by a trusted party strictly following the design specification, can we trust it?
2. When an adversary inserts hardware Trojan into the system, can we detect it?
3. What is the design cost to build an ideal trusted IC, if it exists?

We now elaborate on these questions by the following two illustrative examples. First we consider a 3-input encoder which assigns a 2-bit code to each of the three different objects. There will be three inputs x, y, and z and outputs a and b. The behavior is defined in the following truth table (Table 1).

Using conventional logic minimization techniques, the most efficient circuit that implements this functionality can be expressed as the following logical equations:

$$a = z'$$

$$b = y'$$

Table 2 shows the full functionality defined by the above logical circuit. From the top half of the table, we see that the original desired behavior as specified in Table 1 are met. However, from the bottom half of the table, we see that this logic implementation does more than what is required. For example, code 11 is designed to identify input 100, but input 000 will also generate output code 11; moreover, input 011 or 111 will create an unexpected output code 00. This could be the vulnerability of the design, which an adversary can take advantage to get any output pattern that they want by utilizing the additional input patterns.

One quick solution to this problem is to fully specify the behavior of the desired system. In this example of 3-bit encoder, we can define output 00 as an invalid code and use it for all the five input combinations that are not specified in Table 1. This

Table 2 Truth table defined by the circuit $a = z'$, $b = y'$

	x	y	z	a	b
Original desired behavior	0	0	1	0	1
	0	1	0	1	0
	1	0	0	1	1
Extended unexpected behavior	0	0	0	1	1
	–	1	1	0	0
	1	0	1	0	1
	1	1	0	1	0

leads us to the following circuit implementation:

$$a = x'yz' + xy'z'$$

$$b = x'y'z + xy'z'$$

We have now gone from 2 literals to 12, which, even with logic reuse, will result in a significant increase in hardware overhead and a decrease in the performance of the circuit. This is the cost to make the system trusted.

Next we consider the 3-state finite state machine (FSM) shown in Fig. 1a; we follow the standard procedure to implement this sequential system with two flip flops (FFs) and some logic gates (Fig. 1b). First, from Fig. 1a, we see that when the system is at state B and input is 0, no next state and output are specified, which are known as *don't care transitions*.

However, in the circuit level implementation when FF1 is 0 and FF2 is 1, which reflects the current state B, if input $x = 0$, we can easily verify that FF1 remains unchanged, but FF2 changes to 0. This means that the system switches to state A. At the same time, we can see that the output will be 1. This corresponds to the dashed line from state 01 to state 00 in Fig. 1c. Similarly, state 10 will move to state 00 and output 0 on input $x = 1$.

Second, when both flip flops have value 1, the system will be in a state that is not specified in the original FSM. With input $x = 0$ and $x = 1$, the system will output 1 and move to state C and state A, respectively. In other words, the full functionality of the circuit in Fig. 1b can be described as is shown in Fig. 1c.

The FSM in Fig. 1a is what we want to design, the FSM in Fig. 1c is the system that the circuit we are given (Fig. 1b) actually implements. Clearly we can see the difference between these two FSMs. The one in Fig. 1c has one more state and four more transitions. It is this added state and transitions that creates security and trust concerns for sequential system design.

In the original FSM (Fig. 1a), when the system leaves state A, it cannot come back. In other words, we say that there is no access to state A from state B and state C. However, in the implemented FSM in Fig. 1c, each state has at least one path to come back to state 00. For instance, from state 01 (which is state B in the original FSM), when we inject 0 as the input, the system moves back to 00 (or state A). If state A is a critical state of the system and we want to control its access, FSM in

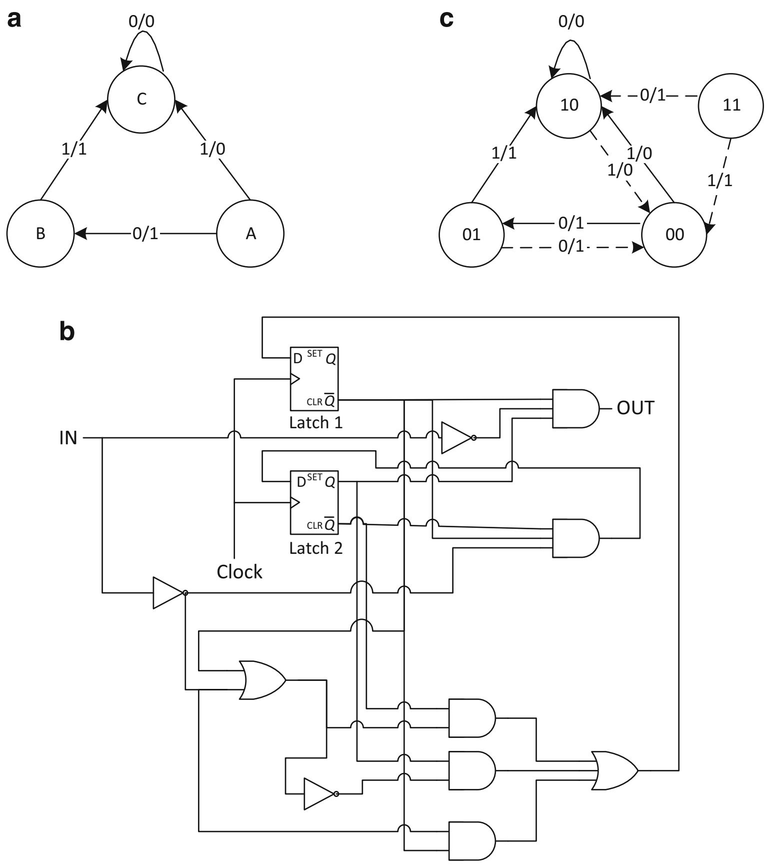


Fig. 1 (**a**) The original 3-state FSM as the system specification. The label on each edge (such as 0/1 from state A to state B) indicates that the transition occurs on input '0' and the transition results in an output '1'. (**b**) The logic/circuit implementation of the 3-state FSM shown in **a**. (**c**) The 4-state FSM generated from the circuit shown in **b** The illustrative example. States A, B, and C in **a** correspond to the states 00, 01, and 10, respectively in **c**. The *dashed* edges are the transitions implemented by the circuit in **b** but not required by the FSM in **a**

Fig. 1a specifies this requirement, but its logic implementation in Fig. 1b violates this access control to state A (or state 00) and the design cannot be trusted.

These examples show that there are security and trust vulnerabilities in the current combinational and sequential system design flow. These vulnerabilities exist in the high level incomplete system specification and become real threats during system implementation. In this chapter we will focus on sequential design and show how an attacker can take advantage of these holes to attack the system. More specifically, we will consider the following two different attacking scenarios.

- **Attacking scenario I:** The attacker attempts to attack a system that is already implemented by exploiting potential trapdoors. The trapdoors in an IC are additional functionalities that are inadvertently implemented during the design, implementation, and fabrication process. They are conceptually different from the hardware Trojans or watermarks which are added intentionally.
- **Attacking scenario II:** The attacker has access to the system during its design and implementation process and can maliciously alter the system specification and/or implementation. This can be considered as adding hardware Trojans as a watermark. However, because the Trojans are introduced early in the system specification and will be integrated into the design, the currently available Trojan detection techniques based on IC abnormality will fail to identify such attacks.

Therefore, it is important to study both attacking scenarios and develop new countermeasures. In this chapter, we will analyze the cost to detect and prevent these attacks. Then we will propose a novel practical method that can be seamlessly integrated into the current sequential system design flow to establish trust in the design and implementation.

The rest of this chapter is organized as follows: in Sect. 2, we give the necessary background of finite state machines. In Sect. 3, we discuss the concept of trust in a finite state machine and its logic implementation. We give necessary and sufficient conditions to build ideal trusted system and system with more relaxed trust metric. Section 4 goes over the vulnerabilities of untrusted designs as well as the benchmark circuits we use in the experimentation. In Sect. 5 we describe two simple attacks that adversaries can utilize to compromise a design and show how powerful such simple attacks can be. Section 6 discusses a naïve countermeasure and the overhead associated with it. Section 7 details our proposed practical approach using flip-flop modifications and the resulting overhead. In Sect. 8, we give the conclusion and Sect. 9 consists of a selection of readings of the related work on trusted IC design, hardware Trojan, and FSM watermarking, which can expand on what we have discussed here.

2 Background of Finite State Machine

A finite state machine (FSM) is defined as a 6-tuple $<I, S, \delta, S_0, O, \lambda>$ where:

I is the input alphabet;
S is the set of states;
δ: $S \times I \rightarrow S$ is the next-state function;
$S_0 \subseteq S$ is the set of initial states;
O is the output alphabet;
λ: $S \times I \rightarrow O$ is the output function.

An FSM can be conveniently represented as a directed weighted (or labeled) graph $G = (V, E)$, where each vertex $v \in V$ represents a state $s \in S$; an edge

$(u, v) \in E$ represents the transition from current state u to its next state v, the weight (or label) on the edge indicates the input–output pair determined by the output function λ. That is, if the edge (u, v) is labeled x/y, then we have $\delta(u, x) = v$ and $\lambda(u, x) = y$ (see Fig. 1a, c for an example). Such graph is often referred to as *state transition graph*.

An FSM is *completely specified* if both the next-state function δ and the output function λ are defined on all possible current state and input pairs (u, x). Otherwise, either δ, λ or both will be undefined on some current state and input pairs (u, x). When δ is undefined, we call this a *don't care transition*; when λ is undefined, its output function has a *don't care*. The FSM is called *incompletely specified* if this happens.

We say that a state v is reachable from state u if and only if there is a directed path from u to v, that is, there is an input sequence following which the state will move from u to v. We define the *reachable set of state u* as

$$R(u) = \left\{ v \in V \middle| \, v \text{ is reachable from } u \right\}$$

which is the set of all states that the system can reach from u and the *starting set of state u* as

$$S(u) = \left\{ v \in V \middle| \, u \text{ is reachable from } v \right\}$$

which is the set of states from which the system can reach u.

For example, in Fig. 1a, $R(A) = \{B, C\}$, $R(B) = \{C\}$, $R(C) = \{C\}$, $S(A) = \phi$, $S(B) = \{A\}$, and $S(C) = \{A, B\}$, where $S(A) = \phi$ because there is no state transition going to state A.

Two the other major concepts utilized in this chapter are *state minimization* and *state encoding*, both used in the process of FSM synthesis. Both of these terms are critical in our proposed method to establish trust in FSM implementation.

Two FSMs are *equivalent* if from the initial state, on any input sequence, they both create the same output sequence. Finding an equivalent FSM with minimal number of states is generally referred as *state minimization* or *state reduction* problem. Figure 2 illustrates this concept, where the two FSMs are equivalent. Intuitively we can see this equivalence from the fact that state B and state D in the 6-state FSM go to the same next state and generate the same output value when they receive the same input. That means that they are a duplicate of each other and we can merge them into one state to reduce the number of states in the FSM. Similarly, once we merge states B and D to one state, we can see that states A and C can also be merged. This results in the reduced 4-state FSM on the right of Fig. 2.

State minimization is an effective approach in logic synthesis to optimize sequential circuit design in terms of area and power. It can be solved optimally in completely specified FSMs and the solution is unique [14]. For incompletely specified machines, although the problem is NP-hard, there are standard approaches to solve the state reduction problem [15].

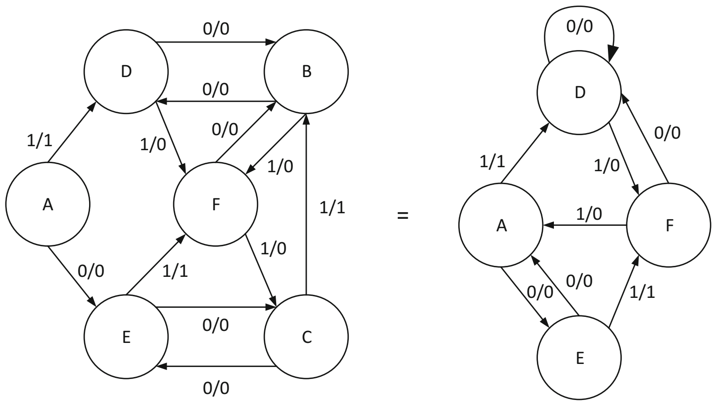


Fig. 2 Two equivalent FSMs. In the 6-state FSM, states B and D (A and C) are equivalent and they are merged to state B (A) as shown in the 4-state FSM

The next phase of FSM synthesis is *state assignment* or *state encoding* where the goal is to assign distinct binary codes to each state of the FSM. Each bit of the binary code will be implemented by one flip flop and the combinational part of the circuitry can be designed to generate input signals for each flip flop and produce the desired output. This will give us a logic implementation of the FSM and concludes the sequential system design. There have been many techniques to solve the state encoding problem based on different optimization objectives (such as area, performance and/or power consumption) and implementation technologies [16].

3 Trusted FSM and Trusted Logic Implementation

Defining metrics for trust is still an open problem. In this section, we will first define the notion of trust for FSM in the most generic way and then use state reachability as an example trust metric to further elaborate the definition of trust. The vulnerabilities, threats, and mitigation methods within the context of FSM design and synthesis will be discussed in the following sections.

When a sequential system is specified as an FSM, from the definition of FSM, we can define trust as follows: ***an FSM is trusted*** *if and only if it makes correct transition from the current state to the next state and produces correct outputs when it received input values.*

From this definition, it is clear that if an FSM is trusted, all of its equivalent FSMs will be trusted, which implies that whether an FSM is trusted or not will not change during the state minimization phase. However, this may change during the state encoding and combinational logic design phase of the FSM synthesis due to the nature of the digital circuit design.

First, digital circuits need to produce deterministic output values based on the logic of the circuitry. This could introduce additional state transitions to the FSM as we have seen from the illustrative example in Fig. 1. In the state transition graph, we can have *don't care* conditions where the next state or the output of the transition or both are not specified. Logic design tools will take advantage of these *don't care* conditions to optimize the design for performance improvement, area reduction, or power efficiency. But when the system (or the FSM) is implemented in the circuit level, these *don't cares* will disappear. The circuit will generate deterministic next states and output for each of these *don't care* conditions. These deterministic values are assigned by CAD tools for optimization purpose and they may make the design untrusted. For example, the next state of a *don't care* transition may be assigned to a state for which access control is required and thus produce an illegal entry to that state (i.e., creating a trapdoor to access that state).

Another reason that an FSM may become untrusted comes from the state encoding phase of the design. When the original FSM has n states after state minimization, it will need a minimum of $k = \lceil log_2(n) \rceil$ bits to encode these states and some encoding schemes that target other design objectives (such as testability and power) may use even longer codes. As we have seen in the illustrative example, when n is not a power of 2, which happens most of the time, those unused codes will introduce additional states into the system, and all transitions from those extra states will be treated as *don't care* transitions during logic synthesis, introducing uncertainty about the trust of the design and implementation of the FSM.

By analyzing the logic implementation of a given FSM M, we can build an FSM M' that captures the behavior of the circuit. We define that ***a logic implementation is trusted*** *if and only FSM M′ is trusted, that is, FSMs M and M′ are equivalent.*

Theorem 1 A sequential system will have a trusted logic implementation from the traditional synthesis flow if and only if

a) the system is completely specified
b) the number of states at the state encoding stage is a power of 2
c) the code used for state encoding is a minimal length code

Proof We first show these three conditions are sufficient. Condition *a)* indicates that there are no *don't care* transitions; conditions *b)* and *c)* guarantee that in the implementation of the system there will not be any additional states or transitions other than those specified in condition *a)*. Therefore, the FSM generated from the logic circuit will be identical (and hence equivalent) to the FSM that specifies the sequential system at the state encoding stage. From the definition, the logic implementation will be trusted.

Now we show all the three conditions are necessary. When condition *a)* fails, an unspecified don't care condition may introduce additional functionality in the logic implementation, making it untrusted. When condition *b)* or *c)* fails, the FSM generated from the logic implementation will have more states then required by the system and the behavior on these extra states will make the logic implementation untrusted. □

An ideal trusted FSM or sequential system can be built if the system satisfies the three conditions in Theorem 1. However, it is unrealistic to assume that these conditions, particularly *a)* and *b)*, will be satisfied. First, given the complexity of today's system with hundreds of input pins and thousands of flip flops, it is impossible to completely specify the system's behavior of all possible input values. Second, there are no effective methods to ensure that the number of state, after state minimization, will be a power of 2. Finally, without any *don't cares* (condition *a)*) and no flexibility in choosing the code length (condition *c)*), the design will be tightly constrained and hard to optimize. This implies that the approach of building trust by eliminating all the don't cares and limiting state encoding will hurt the design quality. We will demonstrate this in later sections.

When we restrict our pursuit of trusted FSM from the ideal one to a certain relaxed trust metric, it is possible to find a practical method to build trusted systems according to such relaxed trust metric. In the rest of this chapter, we will study the trust of FSMs using state reachability as the trust metric.

Considering a given FSM, $M = (V, E)$ (e.g. Fig. 1a), and its logic implementation (e.g. Fig. 1b), let $M' = (V', E')$ be the completely specified FSM generated from the logic implementation of M (e.g. Fig. 1c). Clearly, as graphs, M will be a subgraph of M'. We say that ***the logic implementation of M is trusted*** if and only if for each state $v \in V$ and its corresponding state $v' \in V'$, v and v' have the same reachable sets $R(v) = R(v')$ and the same starting sets $S(v) = S(v')$.

Intuitively, this means that in M', we cannot reach any new states from v $(R(v) = R(v'))$ and no new state can reach v either $(S(v) = S(v'))$. Apparently, the logic implementation in Fig. 1b and the corresponding FSM in Fig. 1c cannot be trusted.

Theorem 2 The following are equivalent definitions for trusted logic implementation: for any state $v \in V$ in an FSM M and its corresponding state $v' \in V'$ in the logic implementation of M,

(1) $R(v) = R(v')$ *and* $S(v) = S(v')$
(2) $R(v) = R(v')$
(3) $S(v) = S(v')$

Proof We need to show that (1) $\Longleftrightarrow$ (2) $\Longleftrightarrow$ (3). Since (1) is the conjunction of (2) and (3), it suffices to show that (2) $\Longleftrightarrow$ (3). We prove (2) $\Rightarrow$ (3) by contradiction as follows. (3) $\Rightarrow$ (2) can be proved similarly.

If (2) holds, but (3) does not, then there must exist a pair of states $v \in V$ and its corresponding state $v' \in V'$ such that $R(v) = R(v')$ but $S(v) \neq S(v')$. That is, we can find a state $u \in S(v)$ but its corresponding state $u' \notin S(v')$ or vice versa. From the definition, we know that $u \in S(v)$ is equivalent to $v \in R(u)$. Hence, if we have $u \in S(v)$ but $u' \notin S(v')$ as we just found, we should also have $v \in R(u)$ but $v' \notin R(u')$, which implies that $R(u) \neq R(u')$, contradicting the assumption that (2) holds. □

We introduce several definitions before we study the vulnerability, threats, and mitigation methods using state reachability as the trust metric. Considering an FSM M, based on whether we want to control the access to a state in M, the states in M can be partitioned into two groups:

- A state v is *protected* if we want to control the access to v in M such that only those states from the starting states $S(v)$ can reach v in M.
- A state v is *normal* or *unprotected* if we do not want to control the access to v in M

For the FSM M' generated from a circuit implementation of the given FSM M, based on the reachability of the state in M', the protected states in M can be either *safe* or *unsafe*:

- A state v is *safe* if v is a protected state in M and v can only be accessed/reached in M' from its starting states $S(v)$. That is, state v is successfully protected in the circuit implementation.
- A state v is *unsafe* if v is a protected state in M but can be accessed in M' by states that do not belong to $S(v)$. That is, the circuit implementation fails to protect the access to state v.

From these definitions, we see that to protect a state v, it is necessary to protect all the states that can reach v. Therefore, in the rest of the discussion, we assume that we will protect both state v and the states in $S(v)$.

An adversary may attempt to gain access to a protected state v from states that are not in $S(v)$. If the adversary succeeds, the state becomes unsafe. Otherwise, all of the states that we want to protect are safe and the implementation of the FSM will be trusted. The goal of trusted sequential system design is to guarantee that the circuit implementation will be trusted.

In the following sections, we will demonstrate that the current design flow fails to protect the protected states (in Sect. 4); it is easy to define attacks to explore this vulnerability and gain access to protected states (in Sect. 5); and we can find mitigation methods to defeat such attacks (in Sects. 6 and 7).

4 Vulnerabilities in Current Design Flow

One of the most important objectives of this work is to demonstrate the vulnerability of traditional FSM synthesis and design flow. Convincing evidence of the existence of such vulnerabilities is needed to motivate and justify our work. We first describe the benchmark circuits and the simulation setups we have developed to validate the findings on the security risks and the effectiveness of our proposed approach.

Table 3 MCNC benchmark circuit information

MCNC circuit index	Circuit name	States	Input bits	Transitions	Edges	Don't cares edges	*Don't care states*
1	bbara	10	4	56	155	5	6
2	bbsse	16	7	53	1800	248	0
3	bbtas	6	2	20	20	4	2
4	beecount	7	3	27	50	6	1
5	dk14	7	3	47	47	9	1
6	dk15	4	3	27	27	5	0
7	dk16	27	2	105	105	3	5
8	dk27	7	1	12	12	2	1
9	dk512	15	1	27	27	3	1
10	ex3	10	2	33	33	7	6
11	ex4	14	6	20	416	480	2
12	ex5	9	2	30	30	6	7
13	ex6	8	5	32	216	40	0
14	ex7	10	2	35	35	5	6
15	keyb	19	7	167	2408	24	13
16	planet	48	7	114	6016	128	16
17	S1488	48	8	250	12, 160	128	16
18	S1494	48	8	249	12, 160	128	16
19	s208	18	11	150	36, 352	512	14
20	sand	32	11	183	63, 552	1984	0
21	sse	16	7	55	1824	224	0
22	styr	30	9	164	15, 296	64	2
23	train11	11	2	24	24	20	5
24	train4	4	2	12	12	4	0

A selection of Microelectronics Center of North Carolina (MCNC) sequential benchmark circuits, shown in Table 3, in their kiss2 finite state machine format, were used. The first column gives the index for each benchmark circuit. The next four columns show, for each circuit, the name, the number of states, the number of input bits, and the number of transitions specified in the blif file [17]. Note that in the blif format, one transition may include multiple input values that move the current state to the next state. For example, if there is a transition from state u to state v on any of the following input values: 1000, 1001, 1010, and 1011, this will be represented in blif format by only one transition with input 10xx.

The last three columns in the table provide information for a more accurate description of each circuit. We first split each transition into edges, where each edge corresponds to only one input value. For example, the above transition on input 10xx will be split into 4 edges. The column "Number of Edges" gives the total number of edges in each circuit. From this, we can easily calculate the "Number of *don't care* edges" shown in the next column. For example, in the circuit 1 (bbara), there are four input bits, which means a total of $2^4 = 16$ different input values. For each of the 10 states, there will be 16 edges, giving a total of 160 edges. The benchmark has 155 edges. So the number of *don't care* edges is $160 - 155 = 5$. The last column gives the number of *don't care* states which can be computed as follows on the same example. We need 4 bits to encode 10 states, but 4 bits will implement 16 states, so the number of *don't care* states is $16 - 10 = 6$.

In most of these FSM benchmarks, each state is reachable from every other state in the FSM. There is no need to protect such states. To produce FSMs with states that we want to be safe, we modify these benchmarks slightly by removing a small amount of transitions from the blif file so that not all of the states are reachable by all other states. In order to edit the FSMs, a program was written to read in a single FSM, remove the transitions one at a time, and record how many states have become unreachable from other states. This process is repeatable and strictly controlled to prevent an excessive number of transitions from being removed so the modified circuit can still reflect the original benchmark. The number of transitions being removed from each circuit is shown in the second column of Table 4. In most

Table 4 FSM modification information

MCNC circuit index	Transitions removed	Edges removed	Unsafe states
1	4	5	2
2	3	56	7
3	4	4	1
4	3	3	0
5	9	9	2
6	5	5	2
7	3	3	0
8	2	2	4
9	3	3	5
10	3	3	7
11	1	32	11
12	2	2	5
13	2	32	2
14	1	1	5
15	3	24	0
16	1	128	19
17	1	128	23
18	1	128	13
19	3	512	0
20	1	1024	31
21	1	32	5
22	2	48	9
23	1	1	3
24	2	2	2

cases, we only remove a couple of transitions. Similar to Table 3, we expand these removed transitions by reporting the number of edges being removed in the next column.

We treat states that are not reachable by some states in the FSM as safe states and consider all other states as normal states. We then use ABC [18] (a public logic synthesis and formal verification tool) to synthesize each of the FSMs to obtain their circuit implementation. Next we analyze these circuit implementations to generate the completely specified FSM. If the safe states now become reachable from any normal states, these states will be considered unsafe. The number of such unsafe states are shown in the last column of Table 4. Note that for circuits 4, 7, 15, and 19, there are no unsafe states, which means that the circuit implementation of these FSMs are trusted. However, the circuit implementations of the rest of the 20 FSMs are all untrusted.

5 Attacks on Untrusted Logic Implementations

With knowledge of the abundant trust vulnerabilities in the systems designed from the current design tools and flow, it leads to the question of how difficult is it to attack a system's logic implementation? Determining this is the second goal of our work. To answer this question, we consider the following two attacking scenarios based on what the adversary can access:

Case I: The adversary can only access the logic implementation of the system or FSM *M'*. The attacking objective is to gain access to the states that are not accessible as specified in the original specification *M*. That is, finding paths, in FSM *M'*, to access certain states that are unreachable in *M*.

Case II: The adversary gets hold of the original system specification, *M*, in the format of FSM and wants to establish a path to reach certain unreachable state. In this case, the attacker can potentially implement such a path into the design. Our challenge, as a result, is to determine how to disguise the secret path.

We describe two naive attacks, one for each case. As we will show in the experimental results section, these two simple attacks turn out to be quite powerful and challenging to defend. Therefore, we do not consider any sophisticated attacks, although they can be developed.

Attack I: The adversary is aware of the vulnerability of the logic implementation of the FSM following the traditional design flow. Therefore, he can launch the "***random walk attack***" (see pseudo code in Fig. 3) and hope to gain access to states that he is not supposed to reach. In this attack, the adversary will try random input sequences. If they lead to the discovery of a protected state (i.e., states that cannot be reached by the adversary according to the design specification), the attack will be successful. This is possible because the FSM synthesis tools will assign values to the *don't care* transitions with the goal

```
1. start from a random state
2. give a random input
3. if (new state == a protected state)
4. { successful attack;
5. break;} //or continue to explore more
   vulnerability
6. else
7. { goto step 2;}
```

Fig. 3 Pseudo code of the random walk attack

of optimizing design objectives such as delay, area, or power. These added transitions may cause some of the safe states to become reachable from states that do not belong to their starting state sets and therefore, causing them to become unsafe.

Attack II: In this attack, the adversary has the original FSM specification of the system before it is synthesized. If he wants to access state v from a state $u \notin S(v)$, the adversary can simply do the following:

- Check whether there is any *don't care* transition from u, if so, he simply makes v as the next state for that transition. This will give him an unauthorized access to state v in the logic implementation of the system.
- If the state transitions from state u are all specified, he can check whether there are any *don't care* transitions from a vertex/state that belongs to $R(u)$, and try to connect that state to v to create a path from u to v.
- If this also fails, then state v in the system is safe with respect to state u in the sense that one can never reach state v from state u. In this case, the attack fails.

Finally, we mention that in case II, the adversary can take advantage of the new states that logic synthesis tools will introduce (that is, when the number of states is not a power of 2 or non-minimal length encoding is used). He can simply launch attack by connecting any of the new states to state v to gain unauthorized access to state v.

Our next goal is to show that, given the vulnerability of the FSM synthesis, how attackers can gain unauthorized access to those unsafe states. For this purpose, we consider the aforementioned "random walk attack", where the attacker randomly starts with a normal state that could not reach the unsafe state in the original FSM. Then random inputs are generated so that the attacker could move around in the completely specified circuit implementation of the FSM. When the attacker reaches the unsafe state, we mark the state as breached. For this experiment, we attempt to breach an unsafe state 10,000 times from a random starting state, and allow the attacker to generate up to 100 random inputs. For each circuit, we choose five unsafe states to attack. If a circuit has less than five unsafe states, we test all of them. Table 5 shows the results of our testing. For all of the rows with "n/a", those circuits do not have any unsafe states. The last column shows a very high breaching rate, 63.28 %

Table 5 Breaching the unsafe states

MCNC circuit index	Average number of breaches (out of 10,000)	Average number of inputs	Average maximal number of inputs	Average minimal number of inputs	Average breach rate
1	913	9.68	58.50	1.50	9.13 %
2	1252.6	10.29	22.80	1.60	12.53 %
3	7457	2.51	17.00	1.00	74.57 %
4	n/a	n/a	n/a	n/a	n/a
5	9779.5	21.15	99.00	1.00	97.80 %
6	10,000	3.18	24.00	1.00	100.00 %
7	n/a	n/a	n/a	n/a	n/a
8	10,000	4.63	31.25	2.00	100.00 %
9	10,000	4.18	33.60	1.60	100.00 %
10	8701	17.21	70.80	1.80	87.01 %
11	9953.2	18.26	98.40	4.60	99.53 %
12	9989.8	8.64	60.20	1.20	99.90 %
13	9998	12.11	94.00	1.00	99.98 %
14	9927	13.10	71.80	1.80	99.27 %
15	n/a	n/a	n/a	n/a	n/a
16	9633.4	22.23	83.60	4.60	96.33 %
17	5.2	6.10	23.20	3.00	0.05 %
18	0	0.00	0.00	0.00	0.00 %
19	n/a	n/a	n/a	n/a	n/a
20	7165.4	42.24	100.00	3.40	71.65 %
21	379.2	1.49	4.80	1.20	3.79 %
22	1224.8	51.72	99.20	4.60	12.25 %
23	2707.67	2.29	11.67	1.33	27.08 %
24	7479.5	2.02	13.50	1.00	74.80 %
Average	6328.31	12.65	50.87	1.96	63.28 %

on average and close to 100 % for almost half of the circuits. The three columns in the middle indicate that among the 10,000 attempts, in the worst case the attacker can succeed with only one or two input values. In all but four benchmarks, the average input length to gain unauthorized access is less than 20.

A malicious designer of a sequential system has access to the original system specification and can add transitions to the blif file before FSM synthesis. (Note that we do not consider the case that the attacker removes or changes transitions from the blif file. In that case, the required functionality of the FSM will not be implemented and such an attack can be detected relatively easily by verification tools.) For an attacker to gain unauthorized access to a safe state while hiding this malicious behavior, the attacker only needs to add one transition. For example, by specifying a previously *don't care* transition from a normal state to a safe state,

Table 6 Area overhead after attacking unsafe states

MCNC circuit index	Baseline area	Overhead of the best malicious design (%)	Average overhead of all malicious designs (%)
1	63, 568	0.0	12.9
2	170, 288	−19.1	−3.7
3	40, 368	−10.3	0.6
4	63, 568	1.5	7.3
5	163, 328	−28.4	−5.5
6	79, 344	−6.4	−0.4
7	263, 552	−2.3	14.4
8	28, 304	0.0	15.9
9	76, 096	−13.4	3.7
10	79, 808	−11.0	−3.1
11	91, 872	−7.1	13.5
12	69, 600	−5.3	13.6
13	167, 040	−9.2	2.9
14	86, 304	−13.4	−0.3
15	284, 432	−17.3	−6.5
16	841, 696	−1.5	14.2
17	880, 208	−8.1	9.8
18	889, 488	−2.2	8.2
19	115, 536	−6.4	3.7
20	779, 056	−10.9	6.3
21	162, 400	−5.4	7.3
22	934, 960	−31.7	−18.9
23	36, 656	−6.3	25.0
24	19, 952	11.6	31.8
Averages		−10.0	5.7

an adversary can change a state's status from safe to unsafe. As we have discussed earlier, a naïve way to prevent such attack is to make the FSM completely specified by specifying all the *don't care* states and the *don't care* transitions.

Tables 6, 7, 8, and 9 report the impact on design quality by this simple attack. We use area, power, maximal negative slack (the difference between the circuit's timing requirement and the longest delay from input to output, which measures how good the design meets its timing requirement), and the sum of negative slack as the metrics for design quality. In all of the tables, the original FSM is synthesized once to give us the baseline for comparison. We assume that the malicious attacker will add only one transition to breach the system. We allow the malicious attacker to add different transitions and design the system ten times. The overhead of best malicious design and the average overhead over all the malicious designs compared with the baseline are reported in the next two columns.

Table 7 Power overhead after attacking unsafe states

MCNC circuit index	Baseline power usage	Overhead of the best malicious design (%)	Average overhead of all malicious designs (%)
1	387	−2.3	25.0
2	1250.2	−21.3	−5.7
3	236.7	−7.6	8.7
4	441.1	−3.9	6.3
5	1211.8	−25.3	−3.6
6	578.5	−3.7	0.2
7	1999.8	0.9	16.2
8	163.1	0.0	27.5
9	557.6	−17.2	2.6
10	592.6	−16.3	−4.8
11	657.3	−6.0	16.2
12	501.4	−9.5	14.9
13	1293.7	−11.3	2.1
14	677	−20.1	−3.9
15	2066.2	−20.5	−7.6
16	6520.6	−1.6	14.1
17	6941.4	−11.1	7.4
18	6969.3	−4.6	6.5
19	790	−6.7	8.3
20	5939.7	−12.2	6.4
21	1177.7	−9.2	7.4
22	7252.3	−32.7	−20.9
23	243.7	−9.7	27.9
24	134.6	−0.2	26.0
Averages		−11.4	6.8

As we can see from these tables, for most of the circuits, the best malicious designs have negative overhead in area, power, and slack, which means that the untrusted circuit implementations have a better design quality when compared to the initial implementation. This is not surprising because this is the best result from many different FSM modifications that the adversary implemented. The impact on design quality by adding one transition may not be dramatic, but if the attacker designs the system multiple times, each time with a slightly different FSM, the synthesis tools may find a design with better quality. For example, in Table 6, there are only two circuits with area increase in the attacker's best design.

However, when we look at the data on the attacker's average design, we see an overhead of about 6 % along each of the quality metrics. This result is very important in several ways. First, it shows that on average, adding even one more transition will incur design overhead; however, the design overhead is so small that

Table 8 Max negative slack overhead after attacking unsafe states

MCNC circuit index	Baseline max slack	Overhead of the best malicious design (%)	Average overhead of all malicious designs (%)
1	6.88	−9.4	8.9
2	12.63	−15.4	5.4
3	4.64	4.1	14.1
4	7.67	−12.4	−0.1
5	13.5	−23.9	−4.6
6	8.92	0.0	6.8
7	19.24	−3.9	6.6
8	3.75	0.0	18.8
9	8.7	−17.8	4.6
10	8.37	−13.1	0.8
11	9.2	−9.3	12.0
12	8.27	−5.2	9.8
13	14.63	−8.3	−0.8
14	8.56	−11.1	5.5
15	17.22	−21.1	−9.6
16	41.57	−2.5	11.7
17	40.25	−9.1	13.2
18	43.04	−9.5	5.1
19	10.44	−1.0	14.4
20	32.04	−12.0	4.2
21	12.66	−6.8	9.2
22	39.02	−24.6	−14.6
23	4.43	−10.4	22.8
24	3.61	22.4	33.7
Averages		−8.8	8.1

such attack cannot be detected by simply evaluating the design quality. Consider the power consumption data in Table 7; only 8 out of the 24 benchmarks have more than 10 % power overhead. The power overhead on other circuits might not be noticeable, and indeed there are power savings on six circuits.

In summary, in 15 of the 24 circuits, the average change to the malicious circuits' statistics (area, slack, and power usage) is within ±10 %. The average design overhead (see the last row of each table) is less than 8 %. Furthermore, such overhead is on the sequential component of the circuit only. If we consider the entire circuit with the combinational circuitry, the overhead will become even smaller. Therefore, it will not be effective to detect malicious design by evaluating the design quality metrics.

Table 9 Sum of max negative slack overhead after attacking unsafe states

MCNC circuit index	Baseline sum of max negative slack	Overhead of the best malicious design (%)	Average overhead of all malicious designs (%)
1	32.05	−3.8	11.7
2	113.19	−15.8	−1.0
3	20.53	−8.5	1.7
4	40.77	−3.1	5.4
5	99.34	−26.0	−8.4
6	53.19	−5.5	0.8
7	124.25	5.9	14.2
8	14.1	0.0	29.2
9	49.34	−20.5	1.5
10	44.25	−22.7	−6.1
11	95.73	−11.0	10.2
12	39.38	−6.1	11.0
13	137.15	−9.7	0.5
14	42.84	−12.0	8.0
15	98.12	−17.3	−7.3
16	966.4	−5.1	9.1
17	919.81	−8.8	13.7
18	970.58	−6.4	6.2
19	56.7	1.9	18.0
20	411.65	−11.4	2.3
21	107.24	−4.3	9.9
22	514.78	−26.5	−15.0
23	15.53	−14.0	31.6
24	10.42	6.8	18.6
Averages		−9.6	6.7

6 A Naïve Countermeasure

The sufficient and necessary conditions in Sect. 3 for a sequential system to be considered trustworthy actually gives the following constructive method to build trusted FSM (illustrated in Fig. 4):

i. perform state reduction to reduce the number of states
ii. add new states $\{s1, s2, \dots, sk\}$ to the FSM such that the total number of states becomes a power of 2
iii. add state transitions $\{s1 \rightarrow s2,\ s2 \rightarrow s3, \dots, sk \rightarrow s1\}$ with *don't care* as the input/output value.
iv. for the other *don't care* transitions, make s1 as their next state
v. use minimal length codes for state encoding

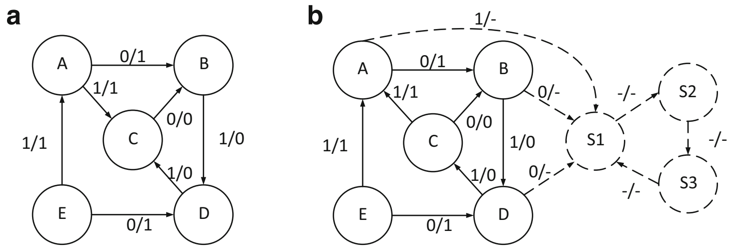


Fig. 4 Implementation of naïve approach. (**a**) Original FSM with unspecified states. (**b**) FSM after applying naïve approach

Apparently, the logic implementation of the FSM following the above procedure satisfies conditions *a)–c)* in Theorem 1 of Sect. 3: step iv ensures that the FSM is completely specified; step ii ensures that the number of states is a power of 2; and step v requires the minimal length encoding.

The only non-trivial part of this procedure is the cycle created in step iii. By doing this, we make these new states not equivalent to each other, and thus prevent the FSM synthesis tools from merging these states. This will ensure that the total number of states is a power of 2.

From the analysis in early part of this section, we know that the FSM built by the above procedure will guarantee a trusted logic implementation of the sequential system. However, such implementation will have very high design overhead in terms of area, power and clock speed. This result can be observed in Table 10.

These results completely reinforce our previous statement that this approach will make the FSM completely specified and over-constrain the design, resulting in designs with potentially very poor quality. For instance, column 4 of Table 10 shows that the average maximal slack has increased by 44 %, which means that the price we pay to ensure trustworthiness in FSM synthesis, by this approach, is probably too high. Such a large slack overhead may not be acceptable for many mission-critical real time embedded systems.

7 A Practical Mitigation Method

To reduce the high design overhead for building trusted FSMs, we propose a novel method that combines modification of gate level circuit and the concept of FSM re-engineering introduced in [19]. Before describing our approach, we mention that to limit the access to a safe state *v*, we need to protect all the states in *v*'s starting set of states. Therefore, in the following discussion, when we consider protecting a state, we consider protecting all of the states in its starting set of states as well.

Table 10 Overhead of naïve countermeasure

MCNC circuit index	Area overhead (%)	Power overhead (%)	Maximum negative slack overhead (%)	Sum of negative slack overhead (%)
1	157.7	202.5	90.4	123.4
2	155.6	147.3	77.7	99.3
3	−6.9	−2.5	6.7	−9.2
4	19.0	21.8	22.3	26.3
5	−20.7	−17.0	−7.3	−13.3
6	85.4	86.8	48.9	82.2
7	33.8	34.2	12.1	30.4
8	52.5	77.1	45.3	85.2
9	26.8	26.0	3.6	16.0
10	79.1	83.7	42.9	63.8
11	116.7	124.5	59.6	60.5
12	76.7	74.3	22.0	62.2
13	130.6	121.2	62.5	69.6
14	67.2	55.3	38.7	59.7
15	34.3	36.8	5.7	28.4
16	68.7	67.2	35.7	39.2
17	50.9	43.4	29.3	20.2
18	46.3	41.9	16.9	24.2
19	214.5	200.4	69.5	94.1
20	84.8	71.8	20.8	32.2
21	221.1	224.9	129.7	170.8
22	11.5	9.5	−2.0	4.5
23	215.2	249.8	168.4	276.8
24	209.3	210.8	73.4	173.0
Averages	89.6	92.8	44.2	66.4

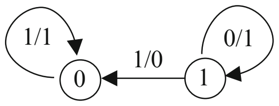


Fig. 5 A simple 2-state FSM

To illustrate the key idea of our approach, we consider a simple 2-state FSM shown in Fig. 5. We assume that state 1 is the safe state and it cannot be reached from state 0. We can add a transition to enforce that the system remains in state 0 on input 0. However, for large design, adding many such transitions will incur high overhead. Instead, we consider how to make state 1 safe at the circuit level. Without loss of generality, we assume that one T flip-flop is used to implement this system. We will use the flip flop content as a feedback signal to control the flip flop input signal (shown as the line with arrowhead in Fig. 6). With the help of this new T flip flop, we see that when the system is at state 1, the feedback signal will not impact the functionality of the flip flop input signal. However, when the system is at the normal state 0, the controlled input signal will disable the T flip flop, preventing the system to go to the safe state 1.

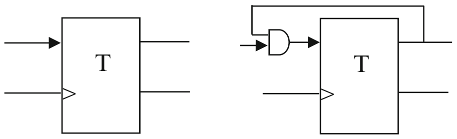


Fig. 6 A normal T flip flop (on the *left*) and a T flip flop with controlled input (on the *right*)

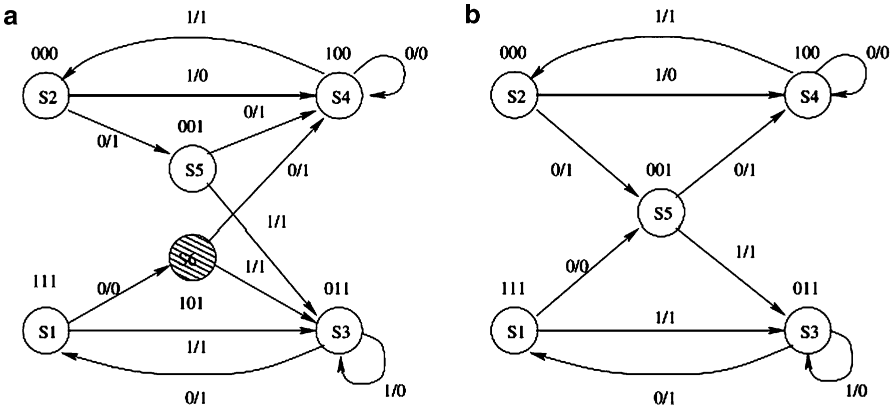


Fig. 7 Example of re-encoded STG through state duplication [19]. (**a**) Original STG. (**b**) Reconstructed STG after duplicating state

Based on this observation, we propose to protect safe states by grouping them and allocating them codes with the same prefix (that is, the leading bits of the codes). For example, when the minimal code length is 5 and there are 4 states to be protected, we can reserve the 4 code words 111XX for these 4 states. Then we use flip flops with controlled signals to implement these prefix (like Fig. 6 shows). The normal states (i.e., states that we do not want to control their access) will have different prefix and thus any attempt of going to a safe state from the normal states will be disabled.

However, when the number of states to be protected is not a power of 2, there will be unused codes with the prefix reserved for safe states. If the synthesis tools assign any of these unused code to other states, these states may gain unauthorized access to the safe states and make them unsafe. To prevent this, we apply the state duplication method proposed in [19] to introduce new states that are functionally equivalent to the safe states (see Fig. 7) and mark them also as states to be protected. We repeat this until the total number of safe state becomes a power of 2.

Figure 7 depicts the concept of state duplication. For state S5 which has two previous states, S1 and S2, and 2 next states S3 and S4 (as shown on the left), we can create a new state S6 and then connecting S6 to all the next states of S5, but partition the previous states of S5 such that some of them go to the new state S6 and others still go to state S5. Clearly, these two FSMs are functionally equivalent. However, by doing this, S5 has a duplicate. If S5 is a state we want to protect, we need to protect its duplicate S6 too. As a result, the number of states to be protected will increase. Consider an FSM with n states, $2^r - 1 < n \leq 2^r$, we apply the state duplication method to extend the total number of states that need to be protected to the nearest power of 2, 2^k. The new FSM will have no more than 2^{r+1} states. So this technique will introduce no more than one additional FF to the design. Accordingly to Yuan et al. [19], the process of state duplication gives us an opportunity to minimize power and/or area. We will report the design quality information (in terms of area, power, and delay) in the next section.

As explained in the previous section, if we do not care about the next state as long as it is not a safe state, we can use flip flops with controlled input to establish trust in the logic implementation of an FSM. To reduce the overhead caused by these controlled input signals, the FSM must be edited. If the number of safe states is less than a power of two, some of the safe states need to be duplicated using a modified version of the process in [19] to duplicate the states that will cause the least, or reduce, the overhead. The next step requires that we partially encode our FSM using a slightly modified version of the encoding algorithm. For example, if we have an FSM with 30 states and 8 states to be protected, all safe states will be given the same partial encoding in the format 11XXX. The rest of the states will have the encodings of 10XXX, 01XXX, or 00XXX. The state encoding algorithm or tool will then fill in the Xs with 0s or 1s in the most efficient manner.

Once this process is complete, the original FSM's states are encoded using the same process but all of the states start with the partial encoding comprised of all Xs. The two FSMs are then compared and the overhead is calculated for the FSM that has been modified for protection of the safe states in comparison to the original FSM. These results are reported in Table 11. The 'encoding bit size' column shows that in most circuits, we will not increase the code length, which means no need of additional flip flops. The 'safe states' column is the percentage of the safe states, which include both the state we want to control the access and their starting set of states. The rest of the five columns report the design overhead in terms of circuit area, gate count, maximum negative slack, sum of negative slack, and power consumption.

The most important result is that in all the design quality metrics, our approach has very limited overhead, from 2.82 % in the maximum negative slack to 7.49 % in the gate count. More specifically, the naïve countermeasure's average overhead on circuit area, most negative slack, sum of negative slack, and power over all the benchmarks are 89.6 %, 44.2 %, 66.4 %, and 92.8 %, respectively (as reported in Table 10). Our new approach can reduce these overhead to 7.01 %, 2.82 %, 5.70 %, and 6.33 %, respectively. Such overhead is about the same or even smaller than the average overhead that a malicious designer will have to suffer (5.7 %, 6.8 %, 8.1 %%, and 6.7 %% as indicated in Tables 6, 7, 8, and 9).

8 Conclusion

Sequential systems are very important components in modern system design. Designing a trusted sequential circuit is crucial to ensure the trust of the overall system. We considered the finite state machine model of sequential circuits and

Table 11 Overhead for manual encoding and state duplication with controlled input to protect safe states

MCNC circuit index	Encoding bit size (%)	Safe states (%)	Gate count (%)	Area (%)	Most negative slack (%)	Sum of negative slack (%)	Power (%)
1	0.00	33.33	6.67	9.49	9.35	15.20	16.15
2	25.00	33.33	6.58	7.71	−2.01	0.08	6.52
3	0.00	0.00	0.00	−1.20	2.30	−5.66	−2.82
4	0.00	33.33	−12.20	−5.17	3.71	−1.39	2.54
5	0.00	33.33	7.14	8.30	2.35	0.37	5.58
6	50.00	33.33	57.50	52.13	32.18	42.11	38.43
7	0.00	0.00	−10.76	−11.17	−9.25	−10.68	−11.27
8	0.00	33.33	0.00	0.00	0.00	0.00	0.00
9	0.00	33.33	−2.63	−2.66	−13.72	−8.57	−5.08
10	0.00	0.00	57.14	53.21	12.69	60.93	49.04
11	0.00	33.33	0.00	0.00	0.00	0.00	0.00
12	0.00	14.29	−19.23	−18.80	−13.02	−23.46	−25.11
13	0.00	33.33	5.68	7.84	7.42	9.55	9.51
14	0.00	0.00	22.86	21.67	21.68	26.20	28.76
15	0.00	33.33	15.13	13.95	14.20	19.40	14.01
16	0.00	39.13	−6.73	−6.01	2.54	2.99	−7.23
17	0.00	60.00	8.21	10.11	16.13	14.13	13.90
18	0.00	60.00	3.12	3.77	4.15	1.97	4.00
19	0.00	0.00	8.70	3.54	−1.77	−3.12	−2.28
20	20.00	6.67	3.91	5.21	−3.66	6.21	4.43
21	25.00	33.33	6.17	7.92	0.56	9.35	5.19
22	20.00	45.45	−22.12	−21.23	−24.38	−22.27	−22.53
23	0.00	33.33	7.14	10.95	7.29	5.97	14.15
24	50.00	33.33	37.50	18.60	−1.11	−2.50	16.05
Average	7.92	27.45	7.49	7.01	2.82	5.70	6.33

defined the notions of trusted sequential system and trusted logic implementation. Then we studied several related trust issues. First, we showed that the current sequential design flow generates systems that can be easily attacked. Then we show a couple of simple and effective methods to attack these designs. Finally, we provided two constructive methods to build trusted logic implementations. The first one is a straightforward method, based on the sufficient and necessary condition for a trusted FSM, but it also introduces a high design overhead. The second approach is based on a simple circuit level modification of the flip flops and can significantly cut the overhead while also guaranteeing the trust of the FSM. We have conducted comprehensive experiments on MCNC benchmarks to validate both our findings about the vulnerability in the current FSM synthesis flow and our proposed approaches to build trust.

We define the notion of trusted FSM both in the most general sense and specifically when state reachability is used as the trust metric. We give a set of necessary and sufficient conditions to build trusted systems in both cases. As the first work of its kind, to the best of our knowledge, we hope that this work can shed lights on how to introduce quantitative metrics for trust in system design.

9 Further Reading

As an extension to our work here, we encourage readers to look into the works presented below. They go further into the topics of trusted design, hardware Trojans, and various security primitives used for improving the trust of a design.

The challenge of building trust in IC is highlighted in a Defense Science Board study on High Performance Microchip Supply, "*Trust cannot be added to integrated circuits after fabrication; electrical testing and reverse engineering cannot be relied on to detect undesired alternations in military integrated circuits*" [1]. The Trusted Foundry Program and the complementary Trusted IC Supplier Accreditation were specifically designed for domestic fabrication, where trust and accreditation are built on reputation and partnership [2]. The notions of trusted and trustworthiness are presented in [3], where the authors discuss the challenges, opportunities, call for new initiatives and programs to establish principles, tools, and standards for building trusted hardware. Trimberger in [4] argues that trusted IC can be built on a field programmable gate array (FPGA) platform because it separates the manufacturing process from the design process. However, the base array still needs to be verified through manufacture and trust still needs to be built during the design process. Suh and Devadas [5] propose physical unclonable functions (PUFs) based on transistor and wire delay to uniquely identify a chip. Logic obfuscation techniques have been proposed recently to solve the IC piracy problem and build trusted IC [6, 7]. However, none of these efforts can be used to verify whether the chip contains unwanted functionalities. Gu et al. [8] develop information hiding method for trusted system design where they impose additional design constraints in the hope that design with unwanted functionalities will incur noticeable performance degradation and thus can be caught. This novelty concept is hard to be implemented due to the complexity of design process.

Hardware Trojan refers to any kind of malicious modification of the IC. It is one of the more serious threats to trusted IC design. Banga and Hsiao [9] propose a circuit partition based approach to detect and locate the embedded Trojan. Rad et al. [10] develop a power supply transient signal analysis method for detecting Trojans based on the analysis of multiple power port signals to determine the smallest detectable Trojan. Wang et al. [11] explore the wide range of malicious alternations of ICs that are possible and propose a general framework for their classification as well as several Trojan detection strategies. Gong and Makkes [12] present a Trojan side-channel based on PUF that can successfully attack block ciphers. Wei et al. [13] develop a set of hardware Trojan benchmarks that are the most challenging representative test cases for side-channel based hardware Trojan detection techniques. The existing hardware Trojan detection approaches rely on catching the misbehavior of the IC caused by the hardware Trojan embedded into a known design. When a hardware Trojan is added during the design process and integrated with the required functionalities of the IC, these approaches will not be effective as pointed out in [8].

There is a rich body of research work on FSM watermarking, for the protection of FSM design intellectual property [20–25]. These techniques usually rely on the modification of the state transition graph at the behavioral synthesis level to embed watermark related to user-specific information for identification purpose. In [20], Oliveira proposes to create watermarks based on a set of redundant states which can only be traversed when a user-specific input sequence is loaded. Lewandowski et al. [21] and Zhang and Chang [22] propose watermarking schemes based on state encoding. Torunoglu and Charbon [23] introduce extra state transitions in the FSM to produce output that carries watermark. Abdel-Hamid et al. [24] improve this method by utilizing the existing transitions for watermarking and successfully reduce the high-overhead caused by extra state transitions. Cui et al. [25] propose an improved scheme that increases the ratio of the number of existing transitions used during the watermarking process to further reduce overhead. The concept behind these FSM watermarking techniques is to embed additional information into the FSM, which is similar to hardware Trojan insertion. However, the added information is for authorship proof and normally does not carry any malicious functionality (like a hardware Trojan does). Another interesting finding about FSM by Yuan and Qu [26] is that during the state minimization stage, many state transitions are redundant in the sense of removing them (i.e., treating them as don't care conditions) will not affect the result of state minimization. They further proposed how to find a maximal set of redundant transitions and manipulate these transitions to hide information. More general information about design intellectual property watermarking can be found in [27].

Acknowledgment The authors were supported by the National Natural Science Foundation of China under Grant No. 61228204, Army Research Office under grant W911NF1210416, Air Force Research Laboratory under agreement number FA8750-13-2-0115, and AFOSR MURI under award number FA9550-14-1-0351. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory or the U.S. Government. The authors were supported by the National Natural Science Foundation of China under Grant No. 61228204, Army Research Office under grant W911NF1210416, Air Force Research Laboratory under agreement number FA8750-13-2-0115, and AFOSR MURI under award number FA9550-14-1-0351. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory or the U.S. Government

References

1. High Performance Microchip Supply: Report of the Defense Science Board Task Force (February 2005). http://www.acq.osd.mil/dsb/reports/ADA435563.pdf
2. Qu, G., Yuan, L.: Design THINGS for the Internet of things: an EDA perspective. In: IEEE/ACM International Conference on Computer-Aided Design (ICCAD'14), pp. 411–416 (November 2014)
3. Irvine, C.E., Levitt, K.: Trusted hardware: can it be trustworthy? In: ACM/IEEE Design Automation Conference, pp. 1–4, June 2007
4. Trimberger, S.: Trusted design in FPGAs. In: ACM/IEEE Design Automation Conference, pp. 5–8, June 2007
5. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: ACM/IEEE Design Automation Conference, pp. 9–12, June 2007
6. Roy, J.A., Koushanfar, F., Markov, I.L.: EPIC: ending piracy of integrated circuits. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 1069–1074, 10–14 March 2008
7. Rajendran, J., Pino, Y., Sinanoglu, O., Karri, R.: Security analysis of logic obfuscation. In: ACM/IEEE Design Automation Conference, pp. 83–89, June 2012
8. Gu, J., Qu, G., Zhou, Q.: Information hiding for trusted system design. In: 46th ACM/IEEE Design Automation Conference (DAC'09), pp. 698–701, July 2009
9. Banga, M., Hsiao, M.S.: A region based approach for the identification of hardware Trojans. In: 1st IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 40–47, June 2008
10. Rad, R., Tehranipoor, M., Plusquellic, J.: Sensitivity analysis to hardware Trojans using power supply transient signals. In: 1st IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 3–7, June 2008
11. Wang, X., Tehranipoor, M., Plusquellic, J.: Detecting malicious inclusions in secure hardware, challenges and solutions. In: 1st IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 15–19 (2008)
12. Gong, Z., Makkes, M.X.: Hardware Trojan side-channels based on physical unclonable functions. WISTP, pp. 294–303 (2011)
13. Wei, S., Li, K., Koushanfar, F., Potkonjak, M.: Hardware Trojan horse benchmark via optimal creation and placement of malicious circuitry. In: ACM/IEEE Design Automation Conference, pp. 90–95, June 2012
14. Hachtel, G., Somenzi, F.: Logic Synthesis and Verification Algorithms. Dordrecht, The Netherlands (1996)
15. Kam, T., et al.: Synthesis of FSMs: Functional Optimization. Dordrecht, The Netherlands (1997)
16. Umans, C., et al.: Complexity of two-level logic minimization. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **25**(7), 1230–1246 (2006)
17. Berkeley Logic Interchange Format (BLIF): http://www.ece.cmu.edu/~ee760/760docs/blif.pdf
18. ABC: A System for Sequential Synthesis and Verification. http://www.eecs.berkeley.edu/~alanmi/abc/
19. Yuan, L., Qu, G., Villa, T., Sangiovanni-Vincentelli, A.: An FSM reengineering approach to sequential circuit synthesis by state splitting. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **27**(6), 1159–1164 (2008)
20. Oliveira, A.L.: Techniques for the creation of digital watermarks in sequential circuit designs. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **20**(9), 1101–1117 (2001)
21. Lewandowski, M., Meana, R., Morrison, M., Katkoori, S.: A novel method for watermarking sequential circuits. In: Proceedings of IEEE International Symposium on Hardware-Oriented Security and Trust, California, pp. 21–24, June 2012
22. Zhang, L., Chang, C.H.: State encoding watermarking for field authentication of sequential circuit intellectual property. In: Proceedings of IEEE International Symposium on Circuits and System, Seoul, pp. 3013–3016, May 2012

23. Torunoglu, I., Charbon, E.: Watermarking-based copyright protection of sequential functions. IEEE J. Solid State Circuits **35**(3), 434–440 (2000)
24. Abdel-Hamid, A.T., Tahar, S., Aboulhamid, E.M.: A public-key watermarking technique for IP designs. In: Proceedings of Design, Automation and Test in Europe, vol. 1, Munich, pp. 330–335, March 2005
25. Cui, A., Chang, C.H., Tahar, S., Abdel-Hamid, A.T.: A robust FSM watermarking scheme for IP protection of sequential circuit design. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **30**(5), 678–690 (2011)
26. Yuan, L., Qu, G.: Information hiding in finite state machine. In: 6th Information Hiding Workshop (IHW'04), vol. 3200, pp. 340–354, LNCS, Springer, New York, May 2004
27. Qu, G., Potkonjak, M.: Intellectual Property Protection in VLSI Designs: Theory and Practice. Dordrecht, The Netherlands (2003). ISBN:1-4020-7320-8

Hardware IP Watermarking and Fingerprinting

Chip-Hong Chang, Miodrag Potkonjak, and Li Zhang

Abstract The continuously increasing gap between silicon and designer productivity has created a need for common design reuse. The initial response to this need emphasized the enforcement of designers' rights through the creation of design such that ownership can be proved with ultra high probability. Our primary objective is to provide sound treatment of the foundations of hardware watermarking and fingerprinting, as well as key research contributions to the field. At the same time, we aim not just to survey and explain key ideas, concepts, and tools but also to identify dominating trends and briefly outline emerging hardware IP protection and, in particular, hardware watermarking and fingerprinting issues including the creation and validation of trusted hardware IP and semantic IP rights protection. Finally, we also elaborate on changing focus from techniques for embedding watermarks and fingerprints to approaches for watermark and fingerprint extraction and on remote digital rights enforcement of hardware IP rights.

1 Introduction

Watermarking of hardware designs is a procedure in which a signature of the designer is embedded into a design in such a way that the design's correct functionality is not impacted and all design metrics are minimally or not impacted at all. Watermarking should be conducted in such a way that watermark extraction is easy while its removal is very difficult in terms of the required design effort and the induced manufacturing and testing cost. In addition, several other requirements may be included such that proving the existence of the embedded signature can be accomplished without revealing it, that a watermarked design can be recognized

C.-H. Chang (✉) • L. Zhang
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore
e-mail: echchang@ntu.edu.sg; lzhang2@e.ntu.edu.sg

M. Potkonjak
Computer Science Department, UCLA, Los Angeles, CA 90095-1596, USA
e-mail: miodrag@cs.ucla.edu

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_10

when it is used within a larger design and that even when a subpart of the watermarked design is used the original author can prove his/her authorship.

The first hardware watermarking efforts were developed in 1997 and reported in early 1998 [1–3]. There were three main sources of inspiration for their development. The first was the emergence of watermarking approaches for audio and video artifacts due to the rapid growth of this market caused by the explosive growth of the Internet. The main idea behind media watermarking is to leverage imperfections in human audio and vision systems so that recorded signals preserve their subjective fidelity while the signatures that indicate the ownership are embedded into them. A great variety of audio, image, video, and text watermarking has been proposed, implemented, and evaluated. Media watermarking has been established as a research area by itself. In particular, a large number of patents have been issued.

The second starting point was the growing gap between silicon and design productivity. The reuse of hardware intellectual property was the best synthesis alternative. It was widely expected that hardware IP blocks will form a viable market. Thus, the Virtual Socket Interface Alliance (VSIA) was formed to coordinate the creation of viable and fast growing hardware IP markers and standards. One of the six main thrusts of VSIA was intellectual property protection (IPP), which resulted in the establishment of the first two hardware watermarking standards.

While the first two impetus sources were market driven, the last one was completely technical. Many synthesis tasks correspond to combinatorial optimization tasks. For example, register assignment in behavioral synthesis and embedded compilation correspond to graph coloring. It has been observed experimentally that for each of these tasks there exist numerous, sometimes even exponentially many, solutions of identical or very similar quality [4–10]. Numerous solutions with optimal and near optimal quality directly enable hardware IP watermarking because the designer can select as his watermarked design a solution with the special property that it serves as proof of legal ownership.

A closely related and extended problem is hardware IP fingerprinting, which is a procedure where a unique tag is created for each chip or a set of ICs. For instance, a design house can create for each of its customers an IP instance with a unique fingerprint for tracing which IP instance is resold by which customer. The main challenge is to avoid very expensive redesign steps for all consequent tasks and to eliminate or at least reduce the number of required unique masks.

The essential difference between watermarking media artifacts and hardware IP is that the latter is a functional entity and its functionality must be fully preserved. In addition, it is important to consider negative impacts on design metrics such as speed and energy. Hence, media watermarking techniques are not applicable. Two main directions have been proposed for hardware watermarking. The first is that additional constraints and/or modification of optimization objectives are imposed in such a way that the quality of the solution is minimally impacted while long and convincing proof of the ownership is guaranteed [11–13]. The second direction is that additional functionality is added as the watermark. For example, a state transition graph is augmented by a specific set of transitions [14, 15], or the unit impulse response of a filter is over-specified in such a way that the author's signature can be observed [16].

The latter approach is interesting because it intrinsically provides for nondestructive and easy extraction of the watermark. That is, in general, a very important issue with hardware watermarking: it is important that remote and easy extraction of watermarking is provided. While hardware reverse engineering techniques are very advanced [17–21], in some applications, nondestructive and remote watermark extraction is essential.

Surprisingly, a common conceptual misconception is that watermarking solutions of synthesis steps is considered to be equivalent to watermarking of corresponding generic graph theoretic problems [22]. It is indeed a standard and widely used technique to map register assignment in behavioral synthesis to graph coloring theoretic frameworks and to use one of many available algorithms [23]. However, once the solution for a corresponding graph coloring problem is incrementally altered with intention of eliminating embedded watermarks, the overall design is also altered. For example, different control logic is required for deciding when which register is accessed as a read or a write. Furthermore, the new control logic requires new interconnects and new control signals from the finite state machine of the overall design. Hence, we need new global and local routing that may impact the overall timing that now has to be validated. In modern designs this may be a very expensive task due to factors such as process variation and crosstalk. Finally, the creation of new masks may be a very expensive proposition. So, for the effective watermarking removal step of graph coloring-based register assignment it is not sufficient to just change registers where variable placement corresponds to assigning corresponding nodes in the corresponding instance of graph coloring. In addition, one has to make sure that consequent synthesis and implementation steps do not have to be altered, which is very often a much more demanding and often impossible requirement to satisfy for designs of any realistic complexity.

Another common related misconception along a similar line of reasoning is that there is not much difference between hardware and software watermarking and that the same techniques can be used interchangeably in the two domains. While this observation is sometimes indeed true and the same concepts are applicable in both domains, there is an essential difference between software and hardware watermarking. However, as we just explained, hardware watermarking is greatly protected due to high synthesis and validation costs of the consequent tasks as well as with the time and economic expense of creating new masks for silicon manufacturing. On another hand additional compilation costs are rather low and often even negligible. Thus, for software watermarking one cannot count on implicit and intrinsic synthesis, validation, and manufacturing costs as effective attack deterrents.

2 Problem Formulation

2.1 *SoC Design Process and IP Core Types*

The SoC design process, as shown in Fig. 1, starts from constructing a system-level model based on the functional specifications of the system. The system-level model consists of interconnected functional blocks, each of which is assigned a specific software or hardware resource. The assignment creates behavioral models of the software and hardware parts of the system. The two parts are then developed separately before they are integrated together in the final stage to form a complete system.

Our discussion focuses on the hardware development. IP cores in different forms are developed and/or adopted in different abstraction levels of the IC design flow.

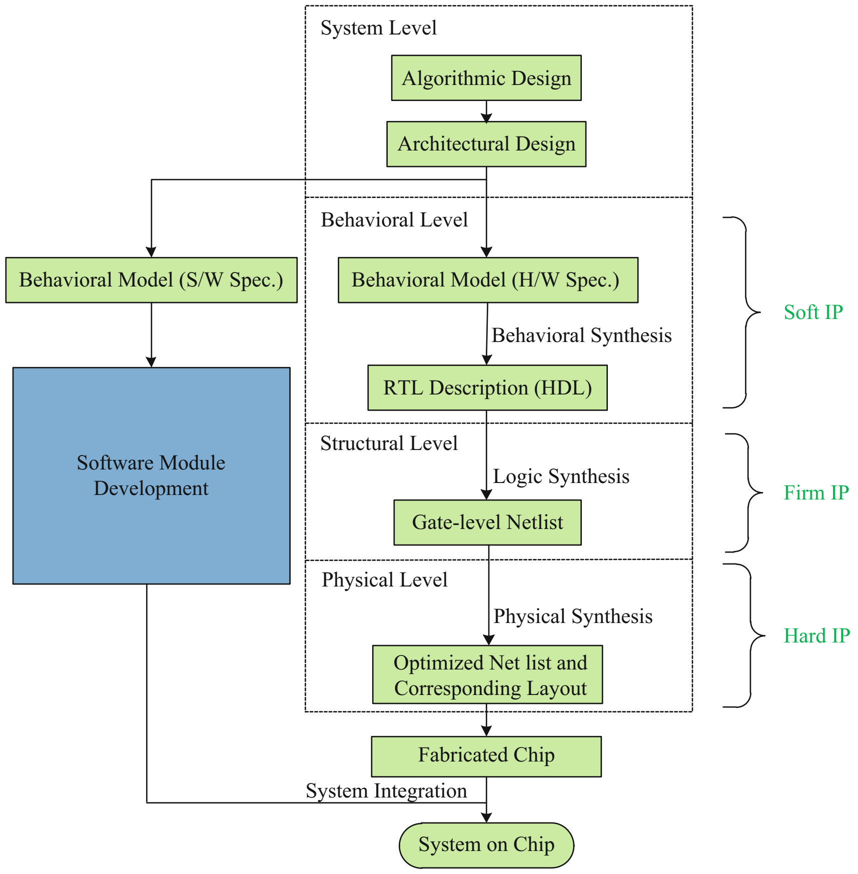


Fig. 1 SoC design flow and three main types of hardware IP cores

According to the Virtual Socket Interface (VSI) Alliance architecture document [24], there are three major types of IP cores, i.e., the soft, firm and hard IP cores.

The soft IP core is used in the behavioral level. These IP cores are usually delivered in the form of synthesizable hardware description language (HDL). This type of IPs provides excellent flexibility to match the requirements of a specific system. The drawback of these IP cores is that their performance is highly dependent on the optimization effort of the system integrator, which is less predictable than the other two types of IP cores.

The hard IP core is used in the physical level. These IP cores target a specific technology and are delivered in the form of fully optimized netlist or the corresponding physical layout. This type of IP cores offers the best performance for the chosen technology library; but due to process dependencies, they have much less flexibility and portability than the soft IP cores. Without requiring any further optimization, the hard IP core is generally released as a drop-in replacement in the physical level design of the system.

The firm IP, or semi-hard IP, refers to IP cores that are in an intermediate form between the soft IP and hard IP. They are usually delivered in the form of gate-level netlist in the structural level. This type of IP cores has more predictable performance than the soft IP and better flexibility and portability than the hard IP. They may also be optimized using a generic technology library, including even physical synthesis steps like floor planning and placement. Nonetheless, no routing is performed and the firm IP remains relatively technology independent. After integrating the firm IP into the system, the system integrator still needs to perform physical synthesis for optimization in a specific technology.

As the soft IP is provided to the system integrator in the form of synthesizable high-level source code, security risk of this IP type is the highest. The high flexibility for reuse also makes infringement of the soft IP very difficult to be detected and traced. In contrast, the limited flexibility of hard IP makes it the easiest to be protected among the three IP types.

The firm IP is sometimes delivered with the synthesizable register-transfer-level (RTL) code for the ease of reuse by the system integrator. In this case, the risk of the firm IP is as high as that of the soft IP. However, if it is transferred alone, i.e., only the gate-level netlist is provided, the firm IP faces the medium risk among the three types of IP cores. The characteristics of the three types of IP cores are summarized in Table 1.

Table 1 Characteristics of the three types of IP cores

	Soft IP	Firm IP	Hard IP
Abstraction level	Behavioral	Structural	Physical
Optimization	Low	Medium	High
Technology dependency	Independent	Generic	Dependent
Flexibility	High	Medium	Low
Distribution risk	High	High (with RTL); Medium (no RTL)	Low

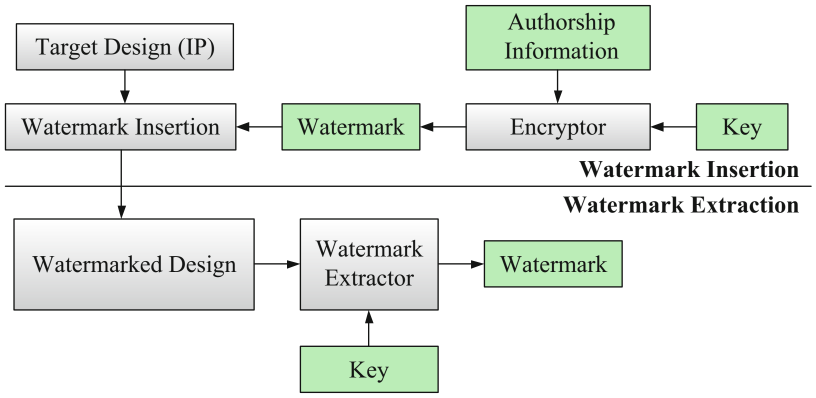


Fig. 2 Generic model for IP watermarking

2.2 Generic Model and Desiderate for IP Watermarking

Similar to watermarking techniques for multimedia applications (e.g., pictures, audio, video or 3D models), hardware IP watermarking is realized by inserting covert and indelible ownership information into the target design for ownership proof. Derived from the generic model for digital watermarking [25], a generic model for hardware IP watermarking is depicted in Fig. 2. The watermark insertion process inserts the watermark into an IP core at some chosen design abstraction level, while the watermark extraction process defines how the watermark is extracted from the watermarked IP core.

The general requirements for IP watermarking are very similar to those of multimedia watermarking. Nonetheless, multimedia watermarking has more freedom to alter the cover media and insert the watermark; it exploits human auditory or visual imperfections to achieve watermark imperceptibility and robustness. Such alteration is restricted in IP watermarking, since the watermarked IP must remain functionally correct. Based on the requirements for an IP watermarking scheme proposed in [26–28], the following desiderata are outlined:

Maintenance of Functional Correctness The functionality of the IP core should not be altered after the insertion of the watermark.

Independence of the Secrecy of Algorithm According to one of the oldest security tenets defined by Kerckhoffs [29], the security of any encryption or security technique lies not in the secrecy of the algorithm, but on the mathematical complexity of such algorithm. Thus, security of the watermark should not depend on the secrecy of the watermark insertion or extraction algorithm but on some system properties.

Strong Authorship Proof The watermarking scheme should be capable of inserting enough data for identification of the IP owner. The data should be of sufficient creditability to be considered as evidence in front of court for proving the authorship.

Low Embedding Cost The embedding of the watermark should be made transparent to existing design processes. The embedding cost, including both the computational cost and time needed for embedding the watermark, should be kept low.

High Reliability The reliability of a watermarking scheme can be evaluated with the robustness and probability of coincidence (P_c) of the watermark. The robustness measures the strength of the hidden signature against various attacks, while the probability of coincidence, sometimes called the false positive rate, is the possibility that the watermark is detected in a non-watermarked design. For non-repudiation, the probability of coincidence should be at least as low as the odd of finding a match fingerprint from two persons in forensic science.

Low Implementation Overhead It is usually inevitable to introduce additional overhead to the IP core after watermarking. The performance overhead, usually measured in terms of area, power and delay, should be kept low for the IP core to remain useful.

Ease of Detection and Tracking Tracking and detection is as important as watermark insertion. It is advantageous to ease the detection of watermark and enable the origin of fraudulence to be traced after possible attacks.

There may be further considerations in designing and evaluating a watermarking scheme. For example, the fairness of the scheme proposed in [7, 30]. A watermarking scheme is fair if it is able to generate watermarked designs at almost the same embedding cost and the same implementation overhead under different authorship signatures. Nonetheless, such a requirement is implicitly implied in the above seven attributes. If a watermarking scheme is not fair, the scheme will have difficulty to find a watermarked solution of high quality with an acceptable cost and effort for an arbitrary signature.

Among the seven attributes, the first six are requirements for the watermark embedding process. This does not mean that watermark extraction is an easy problem. A watermarking scheme is incomplete without a properly designed watermark extraction process. A convenient watermark detection and verification can make a watermarking scheme more practical and receptive. An unmindful watermark detection method could also weaken the robustness of the watermarking scheme.

2.3 Attack Analysis for IP Watermarking

Generally, there are three main types of attacks, i.e., *removal*, *masking* and *forging* attacks. The shared prerequisite of these attacks is that they should not degrade the design quality. That is, an obviously deteriorated design is not what an attacker wants to steal. For removal attacks, the adversary tries to eliminate the watermark completely. This task is usually very hard to succeed with the prerequisite mentioned above. Hence, the attacker may turn to tampering with the watermarked design so that the existence of the watermark cannot be detected, i.e., the masking attack. Depending on the detection mechanism, the minimum percentage of the watermark bits to be altered to result in a successful masking attack varies. A *probability of masking* (P_m) is defined as the probability that an attack would change or delete enough information to render the watermark undetectable without unacceptably deteriorating the performance of the target design [26].

In the forging attack, the adversary embeds his own watermark in the watermarked IP to claim his ownership to the design. To insert the watermark, the attacker may repeat the watermark insertion using his own signature or simply perform a ghost search. A ghost search is an attempt to make up an apparently genuine but different watermark based on the detection method of the targeted watermarked design and use it as the adversary's signature. The probability of a successful ghost search is equal to the *probability of coincidence* mentioned in Sect. 2.2.

Security analysis and countermeasures of an IP watermarking scheme against the removal and masking attacks depend on the employed mechanism for watermark insertion and detection and varies case by case. Nonetheless, there are some common analyses and countermeasures to deal with the forging attack. First, if the watermarked design is forged by merely the addition of watermark, the IP owner is able to provide an IP core with only his watermark while the attacker have only the IP core with both watermarks. It becomes obvious that the IP core belongs to the IP owner. Second, if the attacker has successfully removed or masked the IP owner's watermark and inserted his own, the time of watermark insertion becomes an evidence to distinguish which party is the legal owner. As proposed by Abdel-Hamid et al. [26], a time-stamped authenticated signature can be used. A trusted third party is engaged to act as a watermark certification authority. It is responsible for generating and distributing time-stamped signatures, as well as keeping a record for vouching the authenticity of such signatures for watermark verification in authorship proof. The dispute can be easily resolved by the time stamps of the recovered signatures.

3 Watermark Insertion

In this section, the main methods for watermark insertion in existing hardware IP watermarking schemes are presented. The detection of watermark will be discussed in Sect. 4.

3.1 *Additional Functionality*

An intuitive way to insert the watermark is by adding circuitry to generate the watermark. Fan et al. [31–33] proposed to embed a watermark generating circuit (WGC) into an IP core as a part of the test circuit at the behavioral design level. As shown in Fig. 3, the WGC is composed of some inverters and parallel-input-serial-output (PISO) registers. Based on the test mode signal and inverters, the watermark bits are generated in parallel. These parallel bits are then converted to a serial sequence by the PISO and stored in the shift register. Finally, the watermark sequence is combined with the test patterns in some controllable way by an arbitrator and made detectable in the test output sequence. Depending on the number of output pins used for the output test patterns, several small WGC's may be used to help reducing the overhead of watermarking.

As the WGC and test circuit is inserted to the IP core at the behavioral level and is synthesized with the functional logic, the scheme provides good protection to the IP core at all the lower design levels such as the structural and physical levels. Nonetheless, it is not a good candidate for protecting the soft IP core as the WGC can be easily distinguished at the behavioral level. By simply modifying the number of parallel bits in the WGC, i.e., the watermark length, the strength of authorship

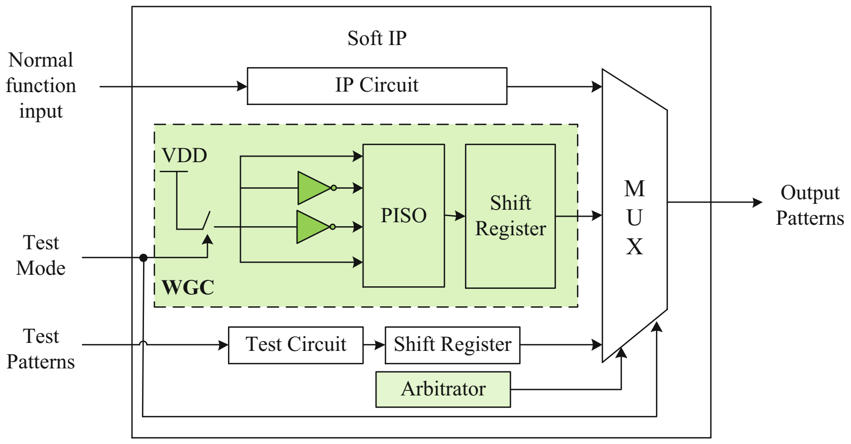


Fig. 3 Architecture of the IP core, watermark generating circuit and test circuit

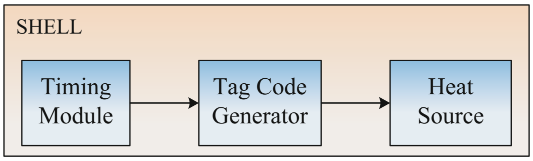


Fig. 4 The tag circuit proposed in [34]

proof can be readily controlled by this scheme. On the other hand, for a fixed WGC, the larger the IP core to be protected, the smaller the overhead due to the WGC.

Kean et al. [34] proposed to insert a small tag circuit, as shown in Fig. 4, into the target design to be protected. The heat source generates heat according to a unique tag signature (i.e., watermark) produced by the code generator, with the timing information from the timing module. The heat source is implemented with a number of parallel ring oscillators. It operates at a high frequency and generates heat when the watermark bit is ‘1’ and turns off at the watermark bit of ‘0’. As a result, the tag circuit in the manufactured chip containing the IP core will generate a sequence of chip temperature changes as the unique tag signature. By measuring the chip temperature data with a thermocouple attached to the chip package and verifying the correlation between the data and tag codes in a database, the tag signature of the IP core can be identified. This is currently the only commercially available tagging approach. To enhance the security, the *Shell* around the tag circuit is fortified by some anti-tamper and reverse engineering countermeasures.

The drawback of measuring the changes in the chip package temperature is the slow data rate due to the physical limitations on how fast a chip package can heat up and how quickly a temperate measurement can be made. Instead of generating temperature changes according to the watermark, Ziener and Teich [35] proposed to convert the watermark to specific power patterns using a power pattern generator. The power pattern generator can be implemented with a set of shift registers and is controlled according to the encoding of the watermark to be transmitted. The power pattern is detected in the reset state of the IP core to avoid interference from the operational logic in the measured power. As some FPGA architectures allow the use of lookup tables (LUTs) as shift register, for FPGA implementations the authors propose to employ some functional LUTs for the shift registers of the power pattern generator. The corresponding watermark embedding process consists of two steps. First, the control logic that is responsible for emitting the authorship signature is merged with the IP core at the HDL level. Second, after logic synthesis, suitable LUTs in the functional logic are selected to implement the shift registers which are attached to the control logic in step 1. Due to the sharing of LUTs, it becomes harder for an attacker to remove or tamper with the shift registers without altering the IP functionality.

Another power based watermarking scheme is proposed by Becker et al. [36]. The tag circuit is also implemented in a high-level description. Instead of trying to detect the power signature directly, the watermark is detected based on correlation as in [34]. The scheme is superior than [35] in that it allows the watermark signal to be hidden below the noise floor of the power side-channel. This makes the watermark detectable even when the IP core is in operation. An additional merit of hiding the watermark below the noise floor is that the watermark is hidden from third parties, making the watermark stealthier. This scheme has the same applicability as all the schemes mentioned above that it is well suited to protect the netlist and designs at lower levels than the HDL IP cores. In the high-level description, it is easy to identify and remove the tag circuit.

3.2 Additional Constraints

In each phase of IC design flow (i.e., the behavioral synthesis, logic synthesis and physical synthesis), there exist a number of NP-hard optimization problems. These problems are too complex to be solved for the exact optimum solutions requiring exhaustive enumeration. Quasi-optimal or near-optimal solutions are sought by heuristic algorithms with some design constraints. This is where constraint-based IP watermarking techniques come into play. The heuristic algorithm takes the design specifications and its performance constraints as inputs for design space exploration to select a good solution as the original IP core from a large solution space. To create a watermarked IP, the encrypted authorship message is first converted into a set of stego constraints. These constraints are then used as either additional inputs to the optimizer (i.e., pre-processing) or imposed on the output of the optimizer (i.e., post-processing). The final result will be a watermarked IP core which satisfies both the original and the stego constraints.

A generic representation of the pre-processing based watermarking procedure is shown in Fig. 5. The watermark is derived from the authorship information based on some encryption processes as depicted in Fig. 2. It is then converted into stego constraints by the constraint generator that directs the mapping from the watermark to the constraints. With the stego constrains added to the inputs of the heuristic solver, the solution space of the *original problem* is reduced to a much smaller solution space of the *stego problem*.

As the synthesis problem for generating the watermarked IP is non-linear and complex, the watermark inserted using the pre-processing approach is usually very robust. This is true especially when it is compared with the watermark inserted in the post-processing approach where the original IP core is first obtained from the synthesis problem and then altered based on the stego constraints. Such alteration may also be exploited by an attacker to mask or remove the watermark. On the other hand, synthesizing the original problem with the stego constraints is more likely to result in unpredictable design overhead. The quality of the watermarked IP cannot be guaranteed even for the optimization intensive watermarking techniques [10]. Hence, the stego constraints in the pre-processing approach must be prudently selected. This problem is not so severe in the post-processing approach. As the stego constraints are imposed on the already optimized solution to the original problem, the overhead due to the inserted watermark can be better controlled.

To limit the overhead and achieve high robustness simultaneously, a three-phase watermarking approach is proposed by Yuan et al. [37]. In the first phase, an optimized solution is obtained from the original synthesis problem. The optimized design is used as a reference in the second phase where some stego constraints are selected by considering the overall tradeoff of solution quality, watermark robustness, etc. In the third phase, the watermarked design is re-synthesized to increase the difficulty for an adversary to tamper with the watermark. As suggested by the authors, the watermark should be inserted close to the end of the synthesis to reduce the complexity of the re-synthesis process in the third phase and to increase the predictability of the change in solution quality due to watermarking.

For all the approaches described above, the strength of the authorship proof depends on the ratio of the stego problem solution space to that of the original problem. The smaller the ratio, the lower the probability of coincidence (because a solution generated under only the original constraints will be less likely to also satisfy the stego constraints), and the stronger the proof of the watermark existence.

The three generic watermark embedding procedures can be applied to various optimization problems in the SoC design process. Among them, the pre-processing approach has the widest appearance. Since its first introduction in [1, 12, 38, 39], an extensive number of constraint-based IP watermarking techniques have been proposed at different abstraction levels, which range from the system synthesis level [1, 6, 12, 13, 40, 41] to the behavioral synthesis level [42–44], logic synthesis level [44–48] and physical synthesis level [12, 44, 49, 50]. In what follows, some of the influential proposals at each level are reviewed.

3.2.1 Constraint-Based IP Watermarking at System Synthesis Level

The system to be designed by the system integrator can be treated as a large monolithic IP core to be protected by inserting a watermark during system synthesis tasks such as multiprocessor DSP code partitioning and cache line coloring.

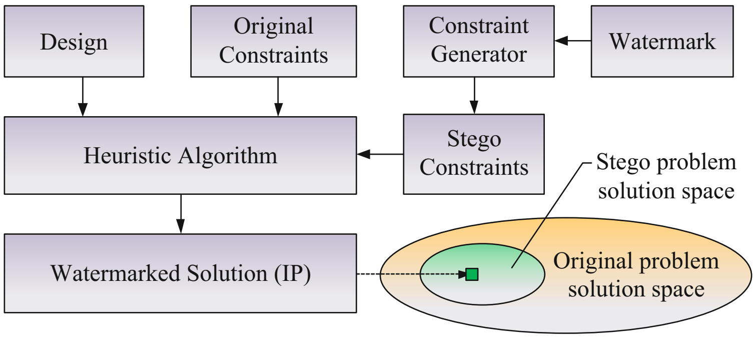


Fig. 5 A generic pre-processing constraint-based IP watermarking procedure

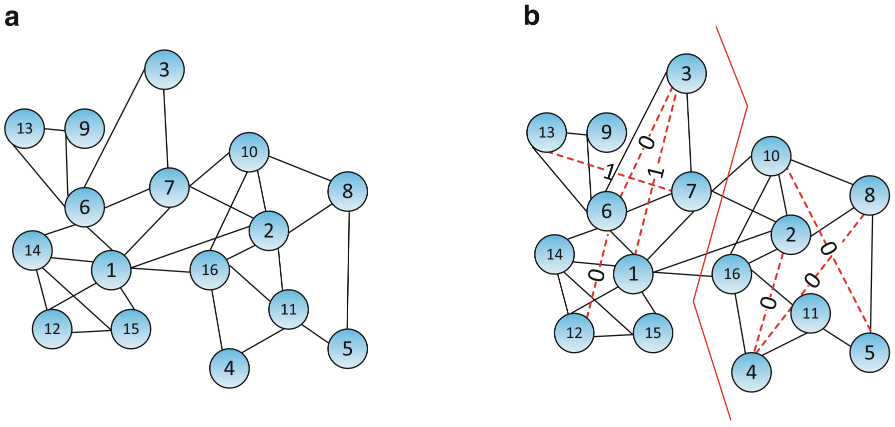


Fig. 6 An example of watermark embedding in the graph partitioning task [1]. (**a**) The graph to be partitioned with nodes indexed by integer numbers. (**b**) The partitioned graph with watermark

Watermarking based on the former task is demonstrated by the example in [1], where the watermark insertion task is formulated as a graph partitioning problem.

In the graph to be partitioned, the nodes are randomly numbered with integers, as shown in Fig. 6a. The stego constraints corresponding to the watermark mandate particular pairs of nodes to remain in the same partition. The following watermark embedding process can be used. For each watermark bit, one origin node and one terminal node are selected for pairing. The original node is selected from those nodes that have not been set as the origin node. The node with the smallest index will be chosen. The determination of the terminal node depends on the value of the watermark bit. When the watermark bit is '1', the node with the smallest odd index that has not been used in the previous pairs will be selected as the terminal node; when the watermark bit is '0', the terminal node will be the node with the smallest even index that has not been paired. Assume the letter 'A' with the ASCII code of "1000001" is a watermark (or a part of the watermark) to be embedded. Based on the embedding criteria described above, the selected pairs of nodes that need to be in the same partition are (1, 3), (2, 4), (3, 6), (4, 8), (5, 10), (6, 12) and (7, 13), each of which is connected with a dotted red line in Fig. 6b. If a balanced partitioning requires the difference in the number of nodes between two partitions to be less than 20 %, the partitioning depicted in Fig. 6b can be obtained, which represents one watermarked solution.

The watermarking scheme described above can be extended to the graph coloring problem. Graph coloring assigns labels, namely "colors", to elements of a graph subject to certain constraints. As a typical form of graph coloring, the aim is to find a way to color the vertices of a graph such that no two adjacent vertices (i.e., connected by an edge) share the same color. Many optimization problems in the VLSI design flow have been modeled as graph coloring problems. For instance, the cache-line code optimization can be solved by finding a solution to the graph coloring problem using a given fixed number of colors, where each color corresponds to one cache line. To insert a watermark into the solution, additional edges corresponding to the watermark can be added into the graph. The watermarked solution will be generated by coloring the new graph using the minimum number of colors.

3.2.2 Constraint-Based IP Watermarking at Behavioral Synthesis Level

In behavioral synthesis, the behavioral model of the design is transformed to a RTL description to implement the behavior. The optimization tasks in behavioral synthesis, such as scheduling, assignment, allocation, transformations and template mapping, are all excellent NP-hard problems for embedding the watermark. The watermarking scheme in [43], which inserts the watermark during register allocation, is used as the example here.

After scheduling the operations with for example, a scheduled control data flow graph (CDFG), the variable values that are generated in one control step and consumed in the later steps must be stored in registers. The interval between the first time a variable is generated and the last time it is used is referred to as its lifetime. Variables whose lifetimes are not overlapped can share the same register, based on which an interval graph can be constructed. In the interval graph, each vertex denotes a variable and the edge between two vertices indicates that there is an overlap in the lifetimes of the two variables. Register allocation can then be performed by solving the graph coloring problem for the interval graph. To insert the watermark, the same approach mentioned in Sect. 3.2.1 can be used. That is, extra edges which represent the stego constraints induced by the watermark (or *watermark constraints* for short) are inserted into the interval graph. Due to the added edges, the resultant watermarked solution satisfies the watermark constraints by storing some specific variables in different registers.

Similar watermark insertion method can also be applied to other behavioral synthesis tasks. The generic watermarking approach is depicted in Fig. 7.

3.2.3 Constraint-Based IP Watermarking at Logic Synthesis Level

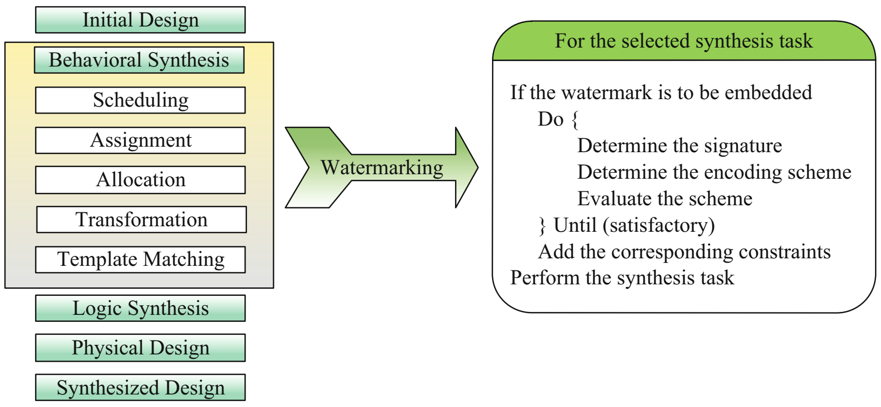
Logic synthesis transforms an abstract form of the design behavior (typically in the RTL HDL) to a specific design implementation constituted by logic gates. Combinational logic synthesis consists of two main optimization tasks, i.e., multi-level logic minimization and technology mapping. Both tasks can be watermarked using the scheme proposed in [45, 46]. We use the example in [46] to illustrate the watermark embedding process for a technology mapping solutions.

Given a cell library, technology mapping (also known as cell-library binding) aims to map the logic network of the design to as few library cells as possible. The complexity of finding an area-optimal solution to the problem is NP-hard [51]. Assume that a cell library consists of two cells as shown in Fig. 8a and the logic network to be mapped is constituted by 11 gates. By performing exhaustive enumeration, we can obtain 49 optimal solutions, each of which uses six cells. The watermark can be embedded by the following procedure.

Firstly, each node (the output of a gate) which is not a primary output is uniquely identified. The set of node identifiers is denoted as $N = \{1, \ldots, 10\}$, as shown in Fig. 8b. The watermark to be embedded is a set of distinct numbers, denoted as W. The cardinality of $|W|$ is required to be smaller than that of $|N|$. Then, the watermarking constraints are imposed by specifying the nodes with identifiers equivalent to numbers in W as pseudo-primary outputs. Due to the watermarking constraints, a specific set of internal nodes will be visible in the resultant solution. If $W = \{3, 4\}$, then node 3 and 4 will be specified as pseudo-primary outputs, as depicted in Fig. 8c. The constrained network can still be solved using only six library cells. However, the number of possible solutions is reduced to 4. This means that the probability of coincidence is $P_c = 4/49$, which represents the strength of the authorship proof. For a real-life design with tens of thousands and more internal nodes, the scheme can achieve very strong proof of authorship with little or no sacrifice on the solution quality.

An alternative scheme that exploits technology mapping for watermark insertion is proposed in [52]. The watermark is inserted by breaking each selected signal node into a pair of primary output and input signals. After technology mapping and a post-processing step where the pairs of added primary output and input signals are shorted, a watermarked technology mapping solution will be generated. By applying the watermarking constraints only to those signal nodes that are not on the critical path, all the critical paths timings are preserved. Again, from the experimental results, the area overhead due to the watermark insertion is very small.

The watermarking scheme depicted in Fig. 8 can be extended to the task of multi-level logic minimization. As proposed in [45, 46], after selected nodes of the logic network are specified as pseudo-primary according to the watermarking constraints,



```
If the watermark is to be embedded
    Do {
        Determine the signature
        Determine the encoding scheme
        Evaluate the scheme
    } Until (satisfactory)
    Add the corresponding constraints
Perform the synthesis task
```

Fig. 7 A generic approach for watermark insertion in behavioral synthesis tasks [43]

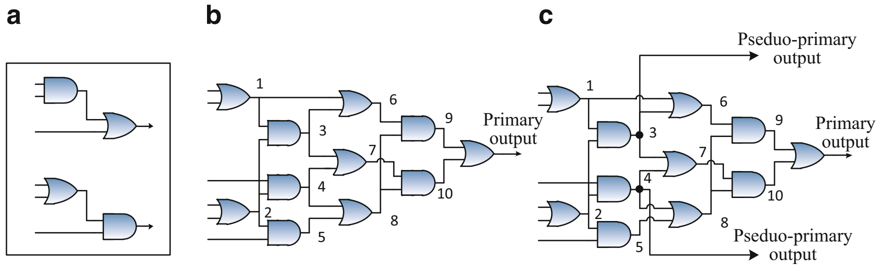


Fig. 8 An example of watermarking technology mapping solutions [45]. (**a**) Cell library. (**b**) A logic network. (**c**) The constrained logic network

an additional logic network will be augmented. The additional logic network takes the pseudo-primary output variables as its input variables and is created according to the copyright-specific pseudo-random bit-stream. The watermarked solution is conceived under the influence of the additional logic network which is to be removed in the synthesis step. The additional logic network has a very significant effect on the sub-function selection. It is almost impossible to produce the same set of sub-functions by an off-the-shelf synthesis tool without knowledge of the augmented copyright-specific logic network, which forms the basis for the authorship proof.

As an additional example, rewiring is an important synthesis technique in logic minimization. It is used for tasks such as logic optimization, delay optimization, elimination of wires with high capacitive loads or switching, etc. A redundancy addition/removal (RAR) based rewiring scheme is proposed in [47]. The basic idea is that a redundant connection may be added to some nodes in the netlist determined by the author's signature without changing the design functionality. A set of redundant connections may be generated and can be removed. The removal of the set of redundant connections makes the added connection irredundant. As a result, the added connections act as the watermark.

3.2.4 Constraint-Based IP Watermarking at Physical Synthesis Level

Physical synthesis begins with a mapped gate-level netlist generated by logic synthesis and outputs a new optimized netlist and the corresponding circuit layout. Common steps for physical synthesis include floorplanning, placement, clock tree synthesis, scan chain generation, routing, etc. Several watermarking schemes based on these tasks are proposed in [12], which are described below.

Watermarking on Path-Timing Constraints

According to the authorship signature, a set of path timing constraints are selected. To insert the watermark, these constraints are replaced with "sub-path" timing constraints. For example, the timing constraint $t\,(C_1 \to C_2 \cdots \to C_{10}) \leq 50$ ns of a path can be replaced with the constraints $t\,(C_1 \to C_2 \cdots \to C_5) \leq 20$ ns and $t\,(C_5 \to C_7 \cdots \to C_{10}) \leq 30$ ns of two sub-paths, where C_i denotes the i^{th} gate in the path. The synthesis solution under the original timing constraint does not necessarily satisfy the two sub-path constraints. According to [12], the chance that satisfying the original constraint happens to satisfy both sub-path constraints is at most one-half. By constraining tens or even hundreds of timing paths in a similar way, a strong proof of authorship will be achieved.

Watermarking on Row-Based Placement
The watermark can also be inserted by constraining the placement of selected logic cells in rows with the specified row parity. That is, some cells are constrained to be placed in an even-index row while some other cells are constrained to be placed in an odd-index row. As a typical placement instance has tens of thousands of standard cells, this approach is capable of inserting a long watermark into the design and providing a high authorship proof. Nonetheless, due to the watermarking constraints, the routing cost between some cells may be increased. Hence, the logic cells that are watermarked needs to be carefully selected.

Watermarking on Standard-Cell Routing
In this approach, the watermarking constraints are the (per-net) costing of the underlying routing resource. That is, if the watermark bit is ‘1’, unusual costs are imposed on “wrong-way” and/or via resources for the selected net, and vice versa. As a result, the watermark can be verified by checking whether a specific set of nets are unusual in their utilization of resources.

3.3 Localized and Hierarchical Watermarking

3.3.1 Localized Watermarking

All the constraint-based watermarking techniques described in Sect. 3.2 are typically realized by encoding the authorship information as a set of stego constraints, augmenting the stego constraints to the original constraints and then optimizing the design from a higher level using an off-the-shelf design tool that finally generates the watermarked IP core. These watermarking techniques employ the pre-processing approach and rely on the generation of a global solution to a design optimization problem according to a specific authorship signature. In these pre-processing based watermarking schemes, the main consideration is usually the watermark robustness and they often gloss over the process of watermark verification. As global optimization is performed for the watermark insertion, the embedded information must be recovered from the entire watermarked IP core indivisibly. A small design alteration by an attacker may cause a substantial distortion in the recovered watermark and result in the failure of watermark detection. The detection difficulty is exacerbated after the protected IP core is integrated into the SoC. In this case, to detect the watermark, the whole IP core needs to be accurately extracted from the system, which is usually very difficult and expensive even with the trapdoor of the IP placement in the SoC.

In view of these problems, Kirovski and Potkonjak [42] proposed to insert multiple *local watermarks* in the IP core to be protected. Instead of creating a large watermark, the authorship signature is converted into a set of smaller watermarks, each of which is randomly augmented into a part of the design and can be verified in its locality independently from the remainder of the design. Two behavioral synthesis tasks are exploited for inserting such watermarks, i.e., operation scheduling and template matching. Basically, both tasks consist of domains with different localities. The small watermarks are then converted into sets of additional constraints and assigned to pseudo-randomly selected domains according to the authorship signature. The scheme enables parts of the IP core to be independently protected because only a segment of the design is needed to decode the stego constraints due to the local watermark in that design segment. Moreover, the local watermarks that are independent from each other force an attacker to alter a substantial part of the IP core in order to obliterate the copyright protection.

Cui et al. [48] proposed a scheme that possesses a similar feature as localized watermarking. They insert the watermark in the technology mapping task of logic synthesis. Unlike the schemes described in Sect. 3.2.3 which require technology mapping to be performed for the whole design to generate the watermarked solution, incremental technology mapping technique is employed to synthesize part of the design for watermark insertion. In particular, a globally optimized technology mapped solution is used as a *master design*. From the master design, the slack sustainability (which determines how well a disjoint closed cone sustains its slack by replacing some of its cells in technology mapping) of disjoint closed cones are estimated. Closed clones with the product of slack and slack sustainability greater than a threshold value are qualified for hosting the watermark bits. Qualified disjoint closed cones are randomly selected and ordered by the authorship signature. The watermark is inserted into these selected closed cones by remapping them according to the constraints imposed by the signature. If the watermark bit to be inserted is '1', the selected closed cone is remapped by coercing the change of one template of the cone; otherwise, the selected template used in the cone is preserved. The watermarked solution is generated by remapping only the selected closed cones according to the stego constraints with incremental synthesis. In fact, the scheme employs the generic three phase watermarking approach described in Sect. 3.2. As the closed cones are qualified based on both slack and slack sustainability over the already optimized master design, it maximizes the embedding capacity and is stealthier than hosting the watermark bits in non-critical paths determined merely by the absolute timing slack. The timing overhead of the watermarked solution can also be kept minimal or even improved.

3.3.2 Hierarchical Watermark

Localized watermarking can be viewed as inserting multiple watermarks in the design at one abstraction level. Instead, Charbon and Torunglu [53, 54] proposed to insert multiple watermarks in different levels, laying the foundation for hierarchical watermarking. With multiple watermarks independently embedded in different abstraction levels, a more robust protection to the IP core is provided. Only when an attacker is able to delete all the watermarks in different design level can he remove the authorship proof. One concern of the hierarchical scheme is the proportionally increase in the watermarking overhead with the number of watermarked levels. This issue can be addressed by a similar idea as localized watermarking in Sect. 3.3.1. The authorship information can be divided into small parts, and independently inserted into every abstraction level to be marked. Besides, as watermarking approaches at different levels have their own strengths and limitations, hierarchical watermarking provides a platform for these techniques to complement each other so that each technique enjoys a greater flexibility and trade-off to achieve a more robust overall IP protection scheme.

The advantage of hierarchical watermarking is exemplified by the watermarking scheme in [55], where the watermark is inserted in both the finite state machine (FSM) and the test scan chain. The FSM watermarking and test architecture watermarking are to be discussed in Sect. 4. The former has good robustness against attacks but the watermark cannot be directly detected after the IP core is integrated into the system. The latter one is vulnerable to attacks but watermark detection is very easy. Thus, the authenticity of the encapsulated FSM IP core can be conveniently detected in the field through the watermark embedded in the scan architecture while the vulnerability of the latter is fortified by the robustness of the FSM watermark. The watermark in the test structure acts like a fragile watermark. Its removal alerts the IP owner to verify the existence of the watermark in the FSM and helps to trace the pirated chips that contain the misappropriated IP core.

4 Watermark Extraction

As discussed in Sect. 2.2, a watermarking scheme is incomplete and turns out to be impractical if it is difficult to detect and track. Except the side-channel watermarking approaches, e.g., the three schemes described in Sect. 3.1, a majority of the watermarking schemes require processing the watermarked solution in order to extract the watermark.

4.1 By Physical Processing

From the watermark extraction point of view, these watermarking schemes can be classified into two types. The first type is the static watermarking scheme, whereby the presence of watermark is verified indirectly by checking if the watermarked constraints generated by the author signature are satisfied. The second type is the dynamic watermarking. For this type of schemes, the watermark can be detected from the output response by injecting a specific input sequence.

4.1.1 Static Detection of Watermark

All the constraint-based watermarking schemes described in Sect. 3.2 are static. To verify the existence of the watermark, the circuit under test (CUT) usually needs to be reverse engineered to the level where the watermarked solution is generated. Then existence of the watermark is proved by checking if the stego constraints due to the authorship signature are satisfied. There are some concerns about such a detection process. First, reverse engineering a CUT to the level where the watermarked solution was originally generated is often a costly task. Second, the verification process usually needs to expose the grammar used for generating the stego constraints from the authorship information. This can potentially weaken the security of other designs that are similarly watermarked.

Some attempts have been made to mitigate the above problems. For example, the localized watermarking described in Sect. 3.3.1 does not require the complete IP core for watermark extraction. Thus the effort and cost of retrieving the watermark will be reduced. Besides, due to the existence of multiple small watermarks in one IP core, even though a subset of the watermarks are exposed in the process of ownership proof, the remaining watermarks are still kept secret. In fact, the watermark verification process in [48] does not even leak the grammar. The process is analogous to the watermark retrieval of non-oblivious image watermarking. Since incremental technology mapping preserves the functionality of the interface ports of remapped cones, the logic functions of these nets can be retrieved from the fan-in and fan-out nodes of closed cones in the master design. To recover a watermarked cone from a marked design, corresponding nodes with the same logic functions are identified. A watermark bit of “0” or “1” can then be determined according to whether the designated template are absent or present in the cone. This watermark retrieval method possesses some degree of fragility that enables the detection of maliciously corrupted watermark bits. When equivalent fan-in and fan-out nodes of a watermarked cone cannot be found, it implies that either cells within the cone or cells surrounding the cone have been modified. Hence, the authorship can be proved by either a perfect or high match between the recovered bit stream and the embedded watermark. Throughout the verification process, the grammar used to generate the watermarking constraints is not exposed.

As an effort to avoid leaking the grammar used to generate the watermarking constraints, Qu [56] proposed a public watermarking approach, where the watermark is divided into a public part and a private part. The private part of the watermark is embedded in a secret way as in the traditional constraint-based watermarking while the public part is embedded in designated locations by methods made known to the public to allow public detectability. This way, detection of the watermark is made easier than that of the conventional schemes but the secret watermark is much well protected. Only when the public part of the watermark is suspected to be attacked will its private portion be recovered for authorship proof. Essentially, this scheme can be deemed as a variant of localized watermarking.

4.1.2 Dynamic Detection of Watermark

Unlike static watermarking schemes, dynamic watermarking does not require the watermarked design to be reverse engineered to the level where the watermark is inserted to perform the watermark detection. Instead, dynamic schemes are characterized by the watermark stimuli-response pair whereby the watermark bits can be detected from the output response by running the watermarked design with a specific input sequence. Such an idea appeared first in [1], where it was proposed to embed a covert channel into a design in such a way that only the authors of a design can observe and interpret information obtained through the channel. The authors illustrate the idea with an example of a digital bandpass filter. The stego constraints which encode a designer signature are added in the filter structure and by observing the outputs for the specific input segments, the embedded message can be identified.

For dynamic watermarking schemes, state transition graphs (STG) of finite state machine (FSM) at behavioral level and test structure such as scan chains at design-for-testability (DfT) level are the two common vehicles. Typical schemes based on these two vehicles are discussed below.

FSM Watermarking

The first FSM watermarking scheme was proposed by Torunoglu and Charbon [57]. This scheme starts with building the FSM representation of the sequential design. Then, unused transitions in the STG are extracted and the watermark is inserted by adding correspondingly defined input/output sequences. In case of completely specified finite state machine (CSFSM), an auxiliary input variable is added to expand the FSM. To satisfy the required strength of authorship proof, the minimum number of transitions needed is calculated and compared with the maximum number of free transitions. If the probability that a non-watermarked design carrying the watermark by coincidence is not low enough, more auxiliary inputs are added. The input sequence is randomized with the set of unused transition inputs to produce an output response that contains the encrypted authorship information. To read the watermark, one should have both the input sequence and the secret key. This scheme works at a high level of the design flow, which provides extra strength. It enables the embedded watermark to be detected dynamically at almost all lower levels, even after design manufacturing. At the same time, the watermark is immune to FSM reduction techniques and is very hard to remove, as the variables used are usually part of other transitions. Nonetheless, this scheme has two deficiencies. Firstly, finding an input sequence that satisfies the required low probability of coincidence and does not incur a high overhead on the STG is an NP-hard problem [15]. Although the authors proposed to use exhaustive search or Monte-Carlo search [58] as the solution, the overhead incurred in the design phase is still high. Secondly, the scheme just makes monotonous use of the unspecified transitions of the STG for watermark insertion. The embedding capacity is limited by the number of free input combinations. If the watermark length is increased beyond the available number of unspecified transitions to boost the authorship proof, the overhead aggravates rapidly.

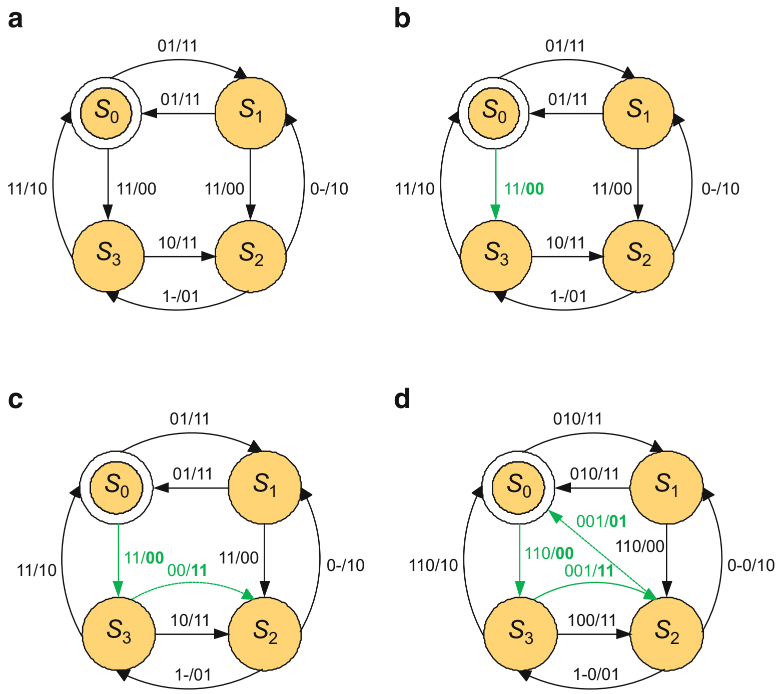


Fig. 9 An example for the watermarking scheme proposed in [59]. (**a**) The original STG. (**b**) The output “00” of S_0 coincides with the first watermark segment. (**c**) One transition added for S_3 using its non-specified input to coincide with the second watermark segment. (**d**) One transition added for S_2 by adding one input bit to coincide with the second watermark segment

To overcome the weaknesses of the previous scheme, Abdel-Hamid et al. [59] proposed to utilize existing transitions as well as unspecified ones in an output mapping algorithm to watermark the FSM, which helps to increase the watermark embedding capacity and lower the overhead. For example, assume a watermark of “001101” is to be inserted into the STG in Fig. 9a. As each transition output consists of two bits, the watermark is partitioned into three segments, i.e., “00”–“11”–“01”. Starting from the stage S_0, a coinciding output “00” is found with the next stage being S_3 (see Fig. 9b). The state S_3 has no output being “11” but it has unspecified inputs. A transition (00/11) from S_3 to S_2 is added into the STG, which yields the STG shown in Fig. 9c. In S_2, there is no output coinciding with the third segment, i.e., “01”, of the watermark, and all inputs are specified. In this case, an extra input bit has to be added to expand the STG. The extra bit is assigned ‘0’ for all existing transitions in the original STG, and ‘1’ for the added transitions. Therefore, a transition (001/01) from S_2 to S_0 is added as the third watermarked transition, as shown in Fig. 9d. When the input sequence “110001001” is injected into the watermarked FSM at state S_0, the watermark can be retrieved from its output sequence.

The embedding process is fast as the watermark bits are inserted at large by a random walk of the STG. When all output bits of an existing transition of a visited node coincide with a substring of the watermark, that transition is employed as part of the watermarking sequence. Otherwise, one of the unspecified transitions is selected to insert some watermark bits. One drawback of this scheme is that coincidence of existing transition's output with the watermark bits becomes rare when the FSM has a high number of output bits. When only unspecified transitions are used for watermarking, this scheme becomes similar to the previous scheme. If there is not enough unspecified transition, pseudo input variables have to be added. Another drawback is the fixed assignment made to the pseudo input variables. This conspicuous discrimination between existing transitions and added transitions may unveil the pseudo-input to an attacker, and the removal of any pseudo input can easily eliminate or corrupt the watermark without affecting the FSM functionality. In addition, the addition of new input variables with fixed assignments on all transitions increases the decoder logics and hence the overhead of watermarked FSM significantly.

A more robust and lower overhead version of STG transition-based watermarking scheme was presented in [60]. To overcome the problem of low probability of coincidence between the watermark substring and existing transition outputs, a longer verification sequence is suggested which results in a response sequence with more bits. In the previous two schemes, the number of watermarked transitions is equal to *m/k*, where *m* is the watermark length and *k* is the number of output bits for each transition. In contrast, the number of possibly watermarked transitions is made larger than *m/k* in this scheme. The localities of the watermark bits are randomly generated and dispersed into the bits of the output sequence. The probability that the designated output bit has the same value as its hosted watermark bit is ½. This tremendously increases the opportunity of employing existing transitions for watermark insertion, reduces the need to add new transitions and consequently reduces the overhead of watermarking. In addition, instead of making the fixed assignment of auxiliary input variables, this scheme keeps their states as don't cares in all transitions except only for the watermarked transitions where their input combinations need to be fixed to host the watermark bits. There are two advantages for this deferred assignment. Firstly, it minimizes the output and next state decoders of the watermarked design. Secondly, the added transitions become indiscernible from the existing ones and removal of the pseudo input variables can affect the FSM functionality. The drawback is that a longer verification sequence and output sequence will be required for watermark retrieval.

The states of the STG can also be utilized for watermark insertion. For example, in [15], the FSM is watermarked by introducing redundancy in the STG so that some exclusively generated circuit properties are exhibited to uniquely identify the IP author. According to the watermark, a specific sequence of states is generated and will only be traversed with the excitation of a specific sequence of inputs. The watermark verification relies on the presence of such extraneous states in the STG. However, the watermark will not survive the removal of all redundant states by state minimization attack. The author provides two possible ways to verify the presence of the watermark, which are the implicit Binary Decision Diagram (BDD)-based enumeration method and the Automatic Test Pattern Generation (ATPG)-based method. The former is too slow for large circuits, whereas the latter requires the solution of an NP-complete problem. It is also not evident that the verification can be carried out efficiently on large circuits.

Test Structure Watermarking
The main limitation of FSM watermarking is that once the watermarked IP is integrated into the chip, the watermarks hidden in the SOC after the chip has been packaged cannot be extracted in the field without dismantling the encapsulation. The only signal that can be traced after chip packaging is the test signal. Thus, numerous post-fabrication verifiable schemes based on the test sequence have been proposed.

In [61], Kirovski and Potkonjak developed a technique for watermarking the design at the logic network level during the selection of the chain of scan registers for sequential logic test generation. The watermark is converted into additional user specific constraints for the selection algorithm. The insertion is realized by using a set of protocols for standardized ordering of the directed graph representation of the circuit. Due to the specific nature of the design of partial scan chains, watermark detection becomes a trivial procedure which is performed by inserting a standard set of test vectors and receiving a set of outputs from the scan chain that is uniquely related to the embedded signature. Similarly, Cui and Chang [62] proposed to add the constraints generated by the owner's digital signature onto the NP-hard problem of ordering the scan cells to achieve a watermarked solution that minimizes the test power. As only the order of scan cells changes, this scheme has no impact on the fault coverage and test application time. The first scheme [33] described in Sect. 3.1 also combines a watermark generating circuit with the test circuit of the IP core at the behavior design level. The watermark sequence is observable from the response patterns in the test mode. All these schemes enable field authentication of the authorship by the IP buyer after the chip has been packaged. However, as the watermarked test core of these schemes is independent of the functional logic, it is not a difficult task for an attacker to redesign the test circuit in order to completely remove or partially corrupt the watermark.

To cope with the limitations, Chang and Cui [63] introduced a synthesis-for-testability (SfT) watermarking scheme. The watermark is embedded as implicit constraints to the scan chain ordering problem as in [62]. Unlike the DfT watermarking methods [33, 61, 62], which are performed after logic synthesis, the SfT watermarking scheme inserts the watermark into the scan chain before logic synthesis. This helps to merge the test functions with the core functions, making the attempts to remove or alter the embedded watermark more costly as such attempts are now quite likely to impact the design specification and optimality. In addition, by using the SfT technique instead of the DfT technique, standard flip-flops can be used as scan register. Without the need to use specialized scan flip-flop (SFF) cells, this scheme is applicable to the protection of IP cores in programmable logic devices. The possible concern of this scheme is that SfT technique is not the mainstream testing technique in industry.

4.2 Side Channel Watermark Extraction

Until recently, efficient watermark detection becomes a rising focus, which results in the emergence of many FSM and test structure based watermarking schemes. As another attempt to simplify the watermark detection, side-channel based watermarking is proposed, where the watermark is detected from side channels such as temperature, power, electromagnetic radiation, etc. Three representatives of such schemes [34–36] have been described in Sect. 3.1. Among them, the scheme by Kean et al. [34] provides the most convenient way to detect the watermark. A thermocouple is attached to the chip package for measuring the temperature changes and the watermark is detected by verifying the correlation between th emeasured data and a database of watermarks. The slow data rate for this temperature based scheme may be a drawback. However, as pointed out by the authors, the time consumed by the measurement is acceptable compared with the alternative of extracting the chip from a system and sending it to a lab for analysis. The measurement process is also much faster than those methods that require electrical contact, considering the time it takes to locate suitable probe-points on the circuit board and connect probes [34].

When multiple watermarked IP cores are integrated together in the system, the emitted side channel signals representing the watermarks of different IP cores will be superposed, leading to interferences among the signals and an increase in complexity to decode the watermarks. This observation holds especially for the power based watermarking scheme in [35] where the power signature is measured directly. To enable the decoding of all the power signatures, the authors proposed a multiplexing method in [64], where the communication channel is divided into multiple logical information channels and one information channel is used for a power signature. The authors explored four different multiplexing methods, i.e., space division multiplexing (e.g., different power pins), time division multiplexing (e.g., different time slots), frequency division multiplexing (e.g., different carrier frequencies), and code division multiplexing (e.g., different decoding codes). The multiplexing methods work best if there is a system-level plan on how the different power signatures in different IP cores are to be multiplexed and decoded. As the IP cores are usually developed by different IP owners, such a requirement may not be easy to satisfy.

The power based side-channel watermarking scheme in [36] is immune to the interferences among multiple power signals of the watermarks. This is because the power signature is detected based on the correlation of the power data and the tag codes stored in a database. The long tag code is pseudo-randomly generated according to the authorship signature. Each tag code is unique if it is sufficiently long. The detection mechanism enables the watermark to be hidden below the noise floor of the power profile. While ensuring that the watermark is easy to be detected in the power side channel, the scheme prevents an arbitrary third party from detecting the watermark.

5 Fingerprinting

The watermarking techniques described above, though capable of identifying the IP ownership, is unable to trace the guilty IP buyer from the unauthorized resold copies. This is because the distributed IP instances to different buyers are identical and carry the same watermark. The problem can be resolved by fingerprinting which embeds a unique and distinguishable mark (i.e., fingerprint) into each distributed IP instance. The malicious buyer who illegally redistributes his IP instance to another entity or misuses his IP instance in an unauthorized application can be easily identified based on the embedded fingerprint.

IP fingerprinting shares very similar desiderata with IP watermarking, such as functionality preservation, high credibility of the proof, low embedding cost and design overhead, high robustness against attacks, transparency to existing design flows, and ease of verification. These desired attributes have been explained in Sect. 2.2. However, the requirements of low embedding cost and high robustness against attacks are more stringent for fingerprinting. The differences are highlighted by examining these attributes in the context of fingerprinting.

Low Embedding Cost A large quantity of distinct high-quality fingerprinted IP instances, instead of the same instance, need to be generated for different buyers. Hence, the incremental embedding cost (mainly the time and effort) for each instance must be kept reasonably low. Caldwell et al. [65, 66] suggest that the run-time for creating every additional fingerprinted instance should be much less than that for solving the synthesis task from scratch. Qu and Potkonjak [8] even advocate that an effective fingerprinting scheme should be able to create $K >> 1$ fingerprinted IP instances at the expense close to that for finding one single solution.

High Robustness Against Attacks The fingerprint needs to be robust against all attacks that can remove or mask a watermark. In addition, the fingerprint must also be collusion-secure. In a fingerprinting scheme, different IP buyers receive different fingerprinted instances. If the inserted fingerprints are the only disparities in different instances, it will be relatively easy for a group of malicious buyers to collude and remove the fingerprint (or more commonly, forge an IP instance from which no buyer in the colluding group can be traced). The collusion attack is especially straightforward if the fingerprint is independent of the functional logic. Hence, Qu and Potkonjak [8] postulate that the IP instance of an innocent buyer cannot be created from any combination of distributed fingerprinted IP instances of a robust fingerprinting scheme and at least one of the guilty buyer can be traced from the forged instance created by a collusion attack. As a rule of thumb, the fingerprint should be closely coupled with the functional logic, making any attempt to remove the fingerprint without affecting the IP functionality or rendering the IP instance useless impossible; the distributed fingerprinted instances should be structurally diverse to disguise the differences incurred by different fingerprints.

It is obvious that the intuitive way to create each fingerprinted instance by repeating the entire process of a typical constraint-based watermarking technique is impractical. While the required engineering time and effort for creating a single

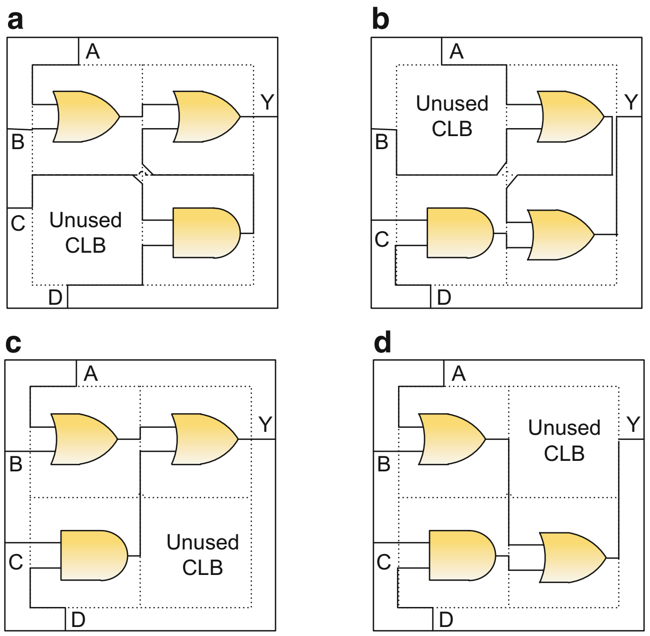


Fig. 10 Four instances (**a**)–(**d**) of the same function in a tile with fixed interface [67]

high-quality watermarked IP instance may be acceptably low, due to the multiplying factor for a large number of IP buyers, the cumulative fingerprinting cost is huge and unrealistic. In fact, only a few proposals [3, 8, 65–68] have managed to reduce the total cost to a reasonable level, which are to be illustrated below.

Lach et al. [3, 67] proposed the first IP fingerprinting scheme for FPGA designs. The FPGA design can be partitioned into small tiles and each tile has several structurally different but functionally equivalent instances. For example, consider a segment of the design for a Boolean function $Y = (A \vee B) \vee (C \wedge D)$. This Boolean function can be implemented in a tile containing four configurable logic blocks (CLB's) and there are four different implementations that are interchangeable as shown in Fig. 10a–d. For all the four implementations, there is one unutilized CLB which can be used to hold a part of the IP buyer's fingerprint (and also the IP owner's watermark). If the same implementation in Fig. 10a is used for all fingerprinted instances distributed to different IP buyers, the difference in the tile among fingerprinted instances will be ascribed only to the inserted fingerprint, and simple comparison collusion will reveal the fingerprint bits. In contrast, by randomly using one of the four implementations for the Boolean function, the difference of various IP buyers' fingerprints will be disguised by the variation in functional structures. Attempts to tamper with the differences revealed by comparing fingerprinted IP instances may alter the functionality of the design and render the design useless.

Due to the employed tiling and partitioning technique, the required effort for generating each fingerprinted instance is greatly reduced. Using the same example depicted in Fig. 10, where the FPGA design is partitioned into four tiles and each tile can be implemented with four different instances, $4^4 = 256$ instances with different functional logic structures can be obtained. Assume that the required effort to place and route an entire design is E, then the effort required for the implementation of each tile instance is around $E/4$. The effort to implement each tile instance, amortized over its use in $4^3 = 64$ different design instances is $E/\ (4 \times 64)$. Hence the effective effort to generate each tiled design is only $E/64$.

As the scheme employs only the unused CLB's in an FPGA design, which always exist, the area overhead is low. However, creating the tiled design and generating functionally equivalent but structurally different instances for each tile may introduce non-negligible timing overhead for the resultant design. The vulnerability of this scheme is that the fingerprint is independent of the functional logic after all. The fingerprint can be removed by reverse engineering the fingerprinted IP instance to an abstraction level before the fingerprint was inserted.

As opposed to the previous scheme which is only applicable to designs with specific regular and highly granular structure, the scheme proposed by Caldwell et al. [65, 66] can be applied to almost all synthesis problems. The basic idea is to obtain an initial seed design by performing the synthesis task from scratch. For each IP buyer, a new fingerprinted instance can be generated with the buyer's fingerprint and the seed design, where only incremental iterative optimization is required. As the effort for creating the seed design is leveraged for generating each fingerprinted design, the scheme manages to reduce the cost of fingerprinting tremendously. Take the standard-cell placement problem for example. The target of this problem is to place each cell of a gate-level netlist onto a legal site with no overlap between two cells and minimized interconnection wire lengths. To create a fingerprinted solution, an initial placement solution S_0 is obtained from scratch. Then, a subset of signal nets N' in the design is selected according to the fingerprint. The weight of each net in N' is set to 10 with the remaining nets set to 1. Based on the solution S_0 and the new net weights, the fingerprinted placement solution can be generated by incrementally replacing the design. The pseudo-code of the fingerprinting process is depicted in Fig. 11. It should be noted that the seed solution for generating the fingerprinted solution S_i may not necessarily be S_0. Instead, the solution S_{i-1}, i.e., the fingerprinted solution for the (i–1)-th user, can be used. As S_{i-1} is a local optimal solution, re-weighting selected signal nets will break the local optimality and facilitate the generation of a new fingerprinted solution S_i that is far away from S_{i-1}. Hence, the new seed design will help to generate fingerprinted instances that are more resilient against collusion attacks. However, as mentioned by the authors, the new solution will inherit the constraints from the previous fingerprinted solution and affect its quality.

It is worth noting that the iterative fingerprinting approach can also be applied to optimization problems that cannot be solved by iterative improvement. For example, given an SAT problem, new solutions cannot be generated from a seed solution by applying iterative improvement techniques. However, a fingerprinted solution can

```
1. Obtain an initial placement solution $S_0$ in LEF/DEF format.
2. For $i$ = 1 to $n$ ($n$ is the number of IP buyers)
3.     Select a subset $N' \subset N$ of the signal nets in the design according to
       the $i$-th user's fingerprint.
4.     Reset the weights of all signal nets to 1.
5.     Set the weight of each net in $N'$ to 10.
6.     Incrementally replace the design, based on the seed solution $S_0$ and
       new net weights, to obtain the fingerprinted placement solution $S_i$.
```

Fig. 11 Pseudo-code of the iterative fingerprinting approach on standard-cell placement

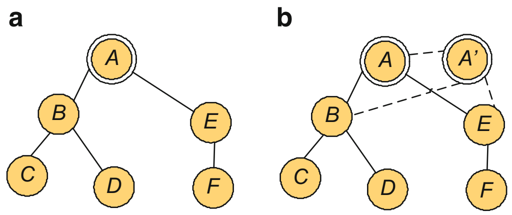


Fig. 12 Duplicating vertex *A* to generate a solution from which two solutions can be derived for the original graph. (**a**) Original graph (**b**) New graph

be generated by solving a new SAT problem with smaller size which is built by preserving the assignments of variables selected according to the fingerprint and removing them (along with their complements) from the initial SAT problem.

In the iterative fingerprinting approach, the incremental optimization effort required to generate each fingerprinted IP instance is still non-trivial. As a large number of fingerprinting instances are usually required, the total amount of effort may still be huge. Qu and Potkonjak [8] proposed a scheme with almost zero incremental effort to generate various fingerprinted solutions after a seed solution is found. The key idea is to introduce a set of independently relaxable constraints before solving the original design problem. From the obtained solution for the modified problem (namely the seed solution), a number of distinct solutions for the original design problem can be derived by independently relaxing each constraint. The authors proved this idea on the NP-complete graph coloring problem. Additional constraints can be introduced by duplicating a selected set of vertices, modifying small cliques or adding edges between unconnected vertices. Take for example the technique of duplicating a selected set of vertices. Assume we have an original graph shown in Fig. 12a. We duplicate vertex A by adding vertex A' and connecting A' with all the neighbors of A. An edge is also added between vertices A and A', resulting in the new graph shown in Fig. 12b. After solving the graph coloring problem for the new graph, a coloring solution can be obtained, with vertices A and A' colored differently. From this solution, two coloring solutions for the original graph can be easily derived with vertex A taking one of the two different colors. Similarly, if k vertices are duplicated and one solution is found for the new graph, a total of 2^k different solutions can be derived for the original graph.

The three techniques mentioned above modify the graph to introduce additional constraints before the graph coloring problem is solved, and hence can be classified as the pre-processing approach. The quality of the derived solutions heavily depends on the added constraints. If the original graph is overly constrained, more colors may be used for the modified graph than the necessary number of colors for the original graph. To cope with this problem, the authors also present a post processing approach. A solution to the graph G (V, E) with k colors is first obtained. Then the vertices of the graph is partitioned into m subsets by their colors with the number of colors in each subset being C_1, C_2, ..., C_m, respectively. The sub-graph formed by the i-th subset of vertices is C_i colorable. As the size of each sub-graph is in general relatively small, all the C_i-color solutions can be found by exhaustive enumeration, denoted as n_i. In this manner, all the solutions for each sub-graph can be exhaustively found and a total number of $n_1 \times n_2 \cdots \times n_m$ solutions can be derived for the graph.

The scheme, although very effective in generating a lot of different solutions with low overhead, lacks a clear mechanism to insert the fingerprint of high credibility. Although each derived solution is guaranteed to be distinct, how distinct the fingerprinted instances are remains a question. If many similarities exist among fingerprinted solutions, the solutions will be vulnerable to collusion attacks.

All the three fingerprinting schemes described above focus on how to create a large number of different quality fingerprinted instances with a low cost and how to make the inserted fingerprint robust against removal, masking and collusion attacks. As in the early-stage watermarking schemes, the ease of fingerprint verification has been neglected. The fingerprint is verified by checking whether the corresponding constraints are satisfied, which is very similar to that of the static watermarking scheme introduced in Sect. 4.1.1. This static type of fingerprint verification process is cumbersome and costly. The verification complexity escalates after the fingerprinted IP is integrated and packaged into a system.

More recently, Chang and Zhang [68] proposed a dynamic fingerprinting scheme that allows the embedded fingerprint to be conveniently detected off-chip. The fingerprint is inserted by constraining the state encoding of a test machine to be embedded into and synthesized with a sequential circuit IP. A test machine [69] is a special FSM that has the following property: An n-FF test machine can be set to any of its state with a corresponding n-bit *synchronizing sequence*; each state of the test machine can be distinguished by applying any n-bit input sequence and observing its output sequence (termed *distinguishing response*). The test machine can be used as an alternative to the scan chain for improving the controllability and observability of sequential circuit IP. As the test machine is synthesized with the design, the fingerprint encoded into the state variables is well integrated with the functional logic and has a global influence on the circuit structure, making the fingerprinted instances inherently collusion-resistant.

The detection of the fingerprint is straightforward. By injecting a specific fingerprint verification sequence, a corresponding distinguishing response can be obtained. The fingerprint bits can then be extracted from the response sequence.

As the fingerprint is detected from the test output, the verification process can be performed conveniently in the field after the fingerprinted IP is integrated in a system.

Typically, the IP owner's watermark and buyer's fingerprint is inserted by concatenation. This naive way doubles the signature length and increases the stego constraints. This issue is first addressed in [68] by using a single fused signature that carries both the watermark and the fingerprint information. As this secure fused signature can be applied to any fingerprinting scheme, its method of generation is described in details below.

The signature is generated through a blind signature protocol [70]. For simplicity, Chaum's blind signature protocol based on RSA is considered. Let (n_A, e_A) and d_A be the certified RSA public and private keys of Buyer A, where $n_A = p_A \cdot q_A$ is the product of two large random primes. The fingerprint F to be embedded into the IP instance for Buyer A can be generated through the following message exchange between the IP provider and Buyer A.

1. *Watermark generation*: The IP provider selects a message to convey its ownership information. The ASCII encoded message is first converted into a binary string and then processed by a hash function such as SHA-2 [71] to generate an integer W, where $0 \le W < n_A$.
2. *Blinding phase*: When Buyer A makes a purchase request, the IP provider randomly selects a secret integer k_A satisfying $0 \le k_A < n_A$ and $gcd(n_A, k_A) = 1$ to compute $W_A = W \cdot (k_A)^{e_A} \mod n_A$. This concealed watermark W_A is then sent to Buyer A.
3. *Signing phase*: To endorse the purchase, Buyer A signs W_A with his secret key d_A and returns his blinded signature $F_A = (W_A)^{d_A} \mod n_A$ to the IP provider.
4. *Un-blinding phase*: By computing $F = F_A \cdot {k_A}^{-1} \mod n_A$, the IP provider obtains the signature of Buyer A on his watermark W, which is the fingerprint to be inserted into the IP instance sold to Buyer A.

The IP provider can verify if the fingerprint F is genuine by decrypting it with Buyer A's public key. Since F is the digital signature of Buyer A on the IP provider's watermark W, it provides an undeniable proof for tracing the redistribution of the copies of IP bought by him. Meantime, as W is concealed in W_A, Buyer A has no knowledge of the IP provider's watermark W and his own signature on W. The blindness of F increases the deterrent effect of uncertainty.

6 Related IP Management and Protection Techniques

There are a large number of IP management and protection techniques that are, to a large or a small extent, related to hardware IP watermarking and fingerprinting. We briefly survey some of them and emphasize their relationship to hardware IP watermarking and fingerprinting.

ID Extraction
The cost of modern transistors is very low, but the cost of modern silicon foundries is very high, even above $1 billion. The manufacturing and testing processes are very complicated and require careful and accurate tracking of each wafer and each integrated circuit. Due to intrinsic process variation each integrated circuit is unique. It has been observed even before the first hardware watermarking and fingerprinting techniques were introduced that it is rather easy to extract a reasonably stable ID for each integrated circuit that can be used for its tracking during the manufacturing and testing processes [72]. There are three key differences between ID extraction and hardware watermarking. The first is that ID extraction is conducted in a secure environment: there is no need to consider attacks. The second is that each chip must have a unique identifier and there is no need for recognizing watermarks. Finally, the extraction of the ID is easy since the chips are still not packaged and are anyhow subject to testing and characterization.

Remote Hardware Enabling and Disabling
There are three essential steps in watermarking-based hardware IPP. By far the best addressed is watermark embedding. Also, it was recognized that watermark extraction is an important hardware IPP task. While initially it was assumed that complete reverse engineering of integrated circuits is a relatively easy task, several techniques later recognized that in many scenarios remote detection of hardware IP is beneficial [73–75]. Also, several hardware watermarking techniques automatically enable easy remote watermark extraction or even make that each output produced by the watermarked circuitry contains information about the watermark [15, 16]. The importance of the third step, watermark enforcement, was recognized more recently. The idea is that only the designer can issue commands for IC activation or deactivation. There are several variants of the basic schemes but all of them share the same IPP mechanisms. Essentially each chip has a physical unclonable function that produces outputs (key) that are required for correct operation of the chip. Only the designer can produce the key after the manufacturer or the owner of the chip provides certain IC measurements [74, 76–81]. While, of course, remote circuit enabling and disabling is more powerful than just watermark embedding as it regularly results in significantly higher hardware and sometimes even timing overheads. The final remark is that recently several techniques have emerged that enable remote software enabling and disabling using hardware-based primitives such as PUFs [82, 83].

Intentional Hardware Trojans
Hardware Trojans are widely studied malicious circuitries that are embedded by attackers in order to enable certain actions that are detrimental to legal owners of the circuits. Typical objectives include extraction of privileged information and targeted denial of service. It has been widely acknowledged that due to process variation and the ultra large scale of integration it is often very difficult to discover hardware Trojans that are embedded either during design or manufacturing of integrated circuits [84, 85]. Interestingly, one can intentionally embedded hardware Trojans in its own designs and chips in such a way that they facilitate hardware

and software IPP [85]. For example, process variation can make only a subset of potential Trojans active. The same results can be achieved if some parts of each chip are intentionally damaged or disconnected. Now, only the manufacturer of the chip knows all non-operating components on a particular chip. So, the relevant compile company may use this information in such a way that each piece of software is compiled in a unique way for each integrated circuit. Hence, software piracy is prevented. Interestingly, intentional hardware Trojans can be realized in such a way that the average and maximal timing degradation is nominal [82, 83].

Hardware Obfuscation
Hardware obfuscation is a technique that implements a piece of digital logic in such a way that it is very difficult and preferably impossible to reverse engineer its functionality. An excellent example of potentially very effective hardware obfuscation approach has been recently proposed and analyzed by researchers at NYU Polytechnic [86]. The key idea is to implement pairs of similar gates in such a way that there is a very small difference in their realization that is difficult for reliable characterization even using destructive reverse engineering techniques. In some sense it employs exactly the same argument that has been used for securing Actel/Microsemi FPGA devices that if one uses a large number of very small fuses, the device is difficult to reliably reverse engineer. The question about resiliency against side channel attacks is currently open. Some recent reverse engineering techniques employ PUFs to implement the logic. Therefore, each chip of the same design has unique structure and unless a small percentage of configurable logic is properly configured, the chip is not functional. Hence the direct manufacturing of copied designs is not possible and the attacker is forced to completely logically reverse engineer the pertinent design. This technique can be easily retargeted in such a way to ensure protection of software intellectual property as well as privacy of data against side channel attacks [82, 83, 87]. The key idea here is to make unique software for each chip of a particular design and/or to encrypt data by encrypting and decrypting it via PUFs.

Hardware Metering
An interesting and important question is how many integrated circuits are produced without permission. For example, this information can be used to figure out what is the size of the monetary damage. Another interesting and somewhat related hardware metering question is how often a particular IP core on a specific chip is used. The first question can be addressed in several ways. For example, one hardware metering technique uses the fish paradox [88] to estimate the number of chip copies [89, 90]. This basic idea can be further augmented to take properties of process variation for more accurate estimation.

Software Metering
Another interesting and economically important question is how many times a particular software program is executed on a specific integrated circuit. This question can be addressed by observing the level of device aging at a set or a system of transistors [90–92]. If there are initial measurements of threshold voltage at these

transistors one can use a simple formula to figure out the aging, i.e. how often the channel of each transistor was under pressure because it was in open switch mode. If there is no such initial measurements, one has to conduct two threshold voltage measurements, one before and one after small intentional aging mechanisms, in order to find the initial position on the aging curve [90]. Note that the same approach can be used for hardware usage metering as defined in the previous short summary.

Foundry Identification
Accurate tracing of the origin of an integrated circuit is an interesting problem with many applications. For example, identifying the source of counterfeited circuits or chips that are produced without proper authorization are economically and strategically important problems. The problem is algorithmically and statistically interesting because now one has to identify a set of statistical properties that are strongly correlated with known chips that are fabricated by a specific silicon foundry. Wendt et al. recently proposed use of Kolmogorov-Smirnov procedure for this task [87].

We finish our brief outline of hardware watermarking and fingerprinting techniques by observing that there is an interesting relationship between them and several other security tasks such as secure remote sensing, secure remote computation, and hardware and software attestation. We indicate in the next section that we believe that the basic concepts of hardware IP watermarking may be used for the creation of qualitatively new security approaches for semantic protection of design intellectual property.

7 Challenges and Opportunities

There are numerous directions that researchers of hardware IP watermarking and fingerprinting are currently pursuing. Maybe the most interesting and important observation is the rapidly growing trend of shifting the initial emphasis from embedding watermarks into design toward their extraction and IP active protection using enabling and disabling. Another interesting trend is towards quantitative evaluation of how many non-authorized ICs are produced and how much they are used.

In this section we focus our attention on three directions that have as their primary starting points on hardware watermarking and fingerprinting techniques that may impact not just hardware security and protection but also software rights enforcement and emerging areas such as big data and the Internet of Things.

Trusted IP Modules
The exponentially growing discrepancy between silicon and designer productivity implies a need for hardware and software reuse. The essential requirement for hardware and software IP is that these functional artifacts are trusted, i.e. they implement their declared functionality and nothing more. In more demanding

scenarios they also should not contain any security vulnerabilities. The creation of trusted hardware and software has attracted surprisingly little attention [93]. The basic idea of this effort is that each IP resource is used in each cycle by the declared functionality such that there exists very little resources and time in which an attacker can launch his attack. The additional focus is placed on the minimization of energy and on low time overhead. Obviously, there is large room and potential in this domain that requires new and fresh ideas and concepts.

Support for Trusted Software Execution
There are only a few actual reported and documented hardware IP attacks. At the same time new software attacks are reported on a daily basis. Only software piracy is estimated on well over $100 billion [94]. Therefore, the importance of software IP protection is of paramount importance. Numerous watermarking and other techniques have been proposed for these tasks [95–97]. Until now, techniques for trusted software protection are implemented after the design and implementation of the architecture is already completed. However, it is clear that some architectures are more vulnerable than others to specific types of attacks. Also, different programs are subject to different attacks that exploit different vulnerabilities. All these constraints can be simultaneously analyzed using techniques from hardware IP watermarking to ensure the creation of secure reusable hardware and software IP.

Semantic Hardware and Software IP Protection
All existing hardware IP protection techniques currently employ only syntax information for defining its objectives and accomplishing its tasks. At the same time, companies are spending hundreds of millions of dollars and sometimes even billions for their patent portfolios and their enforcement. There are also a number of very powerful reverse engineering techniques [20, 21]. There are even a number of techniques that are able to characterize chips using nondestructive techniques and by only employing remotely applied measurements. We believe that the use and processing of semantics can fundamentally alter and greatly improve the effectiveness of IP protection. These goals would also require conceptually new ideas and techniques. Most likely the first semantic IP protection techniques will have narrow objectives and target very specific pieces of semantic knowledge. Semantic hardware and software IP protection as well semantic protection in other domains are very difficult tasks. However, at the same time they are crucial for effective IP protection.

8 Conclusion

Hardware watermarking is a process of creating hardware IP in such a way that created artifacts contain undisputable proofs of authorship. The main flow of hardware IP watermarking consists of watermarking embedding, extraction, and steps for enforcement of digital rights. There are a number of augmenting and optional steps such as watermark enforcement and calculation of properties of the watermarks such as the probability of coincidence, i.e. the likelihood of detection of non-intentional watermarks. We stress that the most critical aspects of hardware watermarking is to establish its global role as a hardware security technique and its relationship with other hardware security tasks. Finally, we proposed several very challenging and potentially practical and economically rewarding research and development directions including the creation of trusted hardware IP and the enforcement of semantic hardware IP rights.

References

1. Hong, I., Potkonjak, M.: Techniques for intellectual property protection of DSP designs. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3133–3136 (1998)
2. Lach, J., Mangione-Smith, W.H., Potkonjak, M: Fingerprinting digital circuits on programmable hardware. In: Information Hiding, pp. 16–31 (1998)
3. Lach, J., Mangione-Smith, W.H., Potkonjak, M.: FPGA fingerprinting techniques for protecting intellectual property. In: IEEE Custom Integrated Circuits Conference, pp. 299–302 (1998)
4. Qu, G., Potkonjak, M.: Analysis of watermarking techniques for graph coloring problem. In: IEEE/ACM International Conference on Computer-Aided Design, pp. 190–193 (1998)
5. Qu, G., Wong, J.L., Potkonjak, M.: Optimization-intensive watermarking techniques for decision problems. In: ACM/IEEE Design Automation Conference, New Orleans, LA, pp. 33–36 (1999)
6. Qu, G., Potkonjak, M.: Hiding signatures in graph coloring solutions. In: Information Hiding, pp. 348–367 (2000)
7. Qu, G., Wong, J.L., Potkonjak, M.: Fair watermarking techniques. In: Asia and South Pacific Design Automation Conference, Yokohama, Japan, pp. 55–60 (2000)
8. Qu, G., Potkonjak, M.: Fingerprinting intellectual property using constraint-addition. In: Design Automation Conference, pp. 587–592 (2000)
9. Qu, G., Potkonjak, M. Intellectual Property Protection in VLSI Design Theory and Practice. Kluwer. ISBN: 1-4020-7320-8 (2003)
10. Wong, J.L., Qu, G., Potkonjak, M.: Optimization-intensive watermarking techniques for decision problems. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **23**(1), 119–127 (2004)
11. Wolfe, G., Wong, J.L., Potkonjak, M.: Watermarking graph partitioning solutions. In: Design Automation Conference, pp. 486–489 (2001)
12. Kahng, A.B., Lach, J., Mangione-Smith, W.H., Mantik, S., Markov, I.L., Potkonjak, M., Tucker, P., Wang, H.J., Wolfe, G.: Constraint-based watermarking techniques for design IP protection. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **20**(10), 1236–1252 (2001)
13. Wolfe, G., Wong, J.L., Potkonjak, M.: Watermarking graph partitioning solutions. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **21**(10), 1196–1204 (2002)
14. Oliveira, A.L.: Robust techniques for watermarking sequential circuit designs. In: Design Automation Conference, pp. 837–842 (1999)
15. Oliveira, A.L.: Techniques for the creation of digital watermarks in sequential circuit designs. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **20**(9), 1101–1117 (2001)
16. Rashid, A., Asher, J., Mangione-Smith, W.H., Potkonjak, M.: Hierarchical watermarking for protection of DSP filter cores. In: IEEE Custom Integrated Circuits Conference, pp. 39–42 (1999)
17. Blythe, S., Fraboni, B., Lall, S., Ahmed, H., Deriu, U.: Layout reconstruction of complex silicon chips. IEEE J. Solid State Circuits **28**(2), 138–145 (1993)

18. Anderson, R., Kuhn, M.: Tamper resistance – a cautionary note. In: The Second Usenix Workshop on Electronic Commerce, pp. 1–11 (1996)
19. Anderson, R., Kuhn, M.: Low cost attacks on tamper resistant devices. In: Security Protocols, Springer, Heidelberg, pp. 125–136 (1998)
20. Torrance, R., James, D.: The state-of-the-art in IC reverse engineering. In: Cryptographic Hardware and Embedded Systems, pp. 363–381 (2009)
21. Torrance, R., James, D.: The state-of-the-art in semiconductor reverse engineering. In: ACM/EDAC/IEEE Design Automation Conference, pp. 333–338 (2011)
22. Van Le, T., Desmedt, Y.: Cryptanalysis of UCLA watermarking schemes for intellectual property protection. In: Information Hiding, pp. 213–225 (2003)
23. Kirovski, D., Potkonjak, M.: Efficient coloring of a large spectrum of graphs. In: Design Automation Conference, pp. 427–432 (1998)
24. VSI Alliance (1997) VSI Alliance Architecture Document: Version 1.0, Santa Clara, CA
25. Cox, I.J., Miller, M.L., Bloom, J.A.: In: Digital watermarking. Morgan Kaufmann Publishers. ISBN: 1-55860-714-5 (2001)
26. Abdel-Hamid, A.T., Tahar, S., Aboulhamid, E.M.: A survey on IP watermarking techniques. Des. Autom. Embed. Syst. **9**(3), 211–227 (2004)
27. Abdel-Hamid, A.T., Tahar, S., Aboulhamid, E.M.: IP watermarking techniques: survey and comparison. In: IEEE International Workshop on System-on-Chip for Real-Time Applications, pp. 60–65 (2003)
28. VSI Alliance (1997) Fall worldwide member meeting: a year of achievement. Santa Clara, CA
29. Kerckhoffs, A.: La Cryptographie Militaire. Journal des sciences militaires. **IX**, 5–38 (1883)
30. Wong, J.L., Majumdar, R., Potkonjak, M.: Fair watermarking using combinatorial isolation lemmas. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **23**(11), 1566–1574 (2004)
31. Fan, Y.C., Yang, H.Y., Tsao, H.W.: Direct access test scheme for IP core protection. In: IEEE Asia-Pacific Conference on Advanced System Integrated Circuits, pp. 262–265 (2004)
32. Fan, Y.C., Tsao, H.W.: Boundary scan test scheme for IP core identification via watermarking. IEICE Trans. Inf. Syst. **E88d**(7), 1397–1400 (2005)
33. Fan, Y.C.: Testing-based watermarking techniques for intellectual-property identification in SOC design. IEEE Trans. Instrum. Meas. **57**(3), 467–479 (2008)
34. Kean, T., McLaren, D., Marsh, C.: Verifying the authenticity of chip designs with the DesignTag system. In: IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 59–64 (2008)
35. Ziener, D., Teich, J.: Power signature watermarking of IP cores for FPGAs. J. Signal Process Syst. Signal Image Video Technol. **51**(1), 123–136 (2008)
36. Becker, G.T., Kasper, M., Moradi, A., Paar, C.: Side-channel based watermarks for integrated circuits. In: IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 13–14 (2010)
37. Yuan, L., Qu, G., Ghouti, L., Bouridane, A.: VLSI design IP protection: solutions, new challenges, and opportunities. In: First NASA/ESA Conference on Adaptive Hardware and Systems, pp. 469–476 (2006)
38. Kahng, A.B., Mantik, S., Markov, I.L., Potkonjak, M., Tucker, P., Huijuan, W., Wolfe, G.: Robust IP watermarking methodologies for physical design. In: Design Automation Conference, pp. 782–787 (1998)
39. Kahng, A.B., Kirovski, D., Mantik, S., Potkonjak, M., Wong, J.L.: Copy detection for intellectual property protection of VLSI designs. In: IEEE/ACM International Conference on Computer-Aided Design, pp. 600–604 (1999)
40. Chapman, R., Durrani, T.S.: IP protection of DSP algorithms for system on chip implementation. IEEE Trans. Signal Processing **48**(3), 854–861 (2000)
41. Megerian, S., Drinic, M., Potkonjak, M.: Watermarking integer linear programming solutions. In: Design Automation Conference (DAC), pp. 8–13 (2002)
42. Kirovski, D., Potkonjak, M.: Local watermarks: methodology and application to behavioral synthesis. IEEE Trans. CAD **22**(9), 1277–1284 (2003)

43. Koushanfar, F., Hong, I.K., Potkonjak, M.: Behavioral synthesis techniques for intellectual property protection. ACM Trans. Des. Automat. Electr. Syst. **10**(3), 523–545 (2005)
44. Charbon, E., Torunoglu, I.: Watermarking techniques for electronic circuit design. In: Lecture Notes on Computer Science 2613, pp. 147–169 (2000)
45. Kirovski, D., Hwang, Y.Y., Potkonjak, M., Cong, J.: Intellectual property protection by watermarking combinational logic synthesis solutions. In: IEEE/ACM International Conference on Computer-Aided Design, pp. 194–198 (1998)
46. Kirovski, D., Hwang, Y.Y., Potkonjak, M., Cong, J.: Protecting combinational logic synthesis solutions. IEEE Trans. CAD **25**(12), 2687–2696 (2006)
47. Khan, M.M., Tragoudas, S.: Rewiring for watermarking digital circuit netlists. IEEE Trans. CAD **24**(7), 1132–1137 (2005)
48. Cui, A.J., Chang, C.H., Tahar, S.: IP watermarking using incremental technology mapping at logic synthesis level. IEEE Trans. CAD **27**(9), 1565–1570 (2008)
49. Charbon, E., Torunoglu, I.: Watermarking layout topologies. In: ASP-DAC, pp. 213–216 (1999)
50. Narayan, N., Newbould, R.D., Carothers, J.D., Rodriguez, J.J., Holman, W.T.: IP protection for VLSI designs via watermarking of routes. In: IEEE International Asic/SoC Conference, pp. 406–410 (2001)
51. De Micheli, G.: Synthesis and Optimization of Digital Circuits. McGraw-Hill, New York (1994). ISBN 0070163332
52. Meguerdichian, S., Potkonjak, M.: Watermarking while preserving the critical path. In: Design Automation Conference (DAC), pp. 108–111 (2000)
53. Charbon, E.: Hierarchical watermarking in IC design. In: IEEE Custom Integrated Circuits Conference, pp. 295–298 (1998)
54. Charbon, E., Torunoglu, I.: Intellectual property protection via hierarchical watermarking. In: International Workshop on IP Based Synthesis and System Design (1998)
55. Cui, A.J., Chang, C.H., Zhang, L.: A hybrid watermarking scheme for sequential functions. In: IEEE International Symposium on Circuits and Systems, pp. 2333–2336 (2011)
56. Qu, G.: Publicly detectable watermarking for intellectual property authentication in VLSI design. IEEE Trans. CAD **21**(11), 1363–1368 (2002)
57. Torunoglu, I., Charbon, E.: Watermarking-based copyright protection of sequential functions. IEEE J. Solid State Circuits **35**(3), 434–440 (2000)
58. Cashwell, E.D., Everett, C.J.: A Practical Manual on the Monte Carlo Method for Random Walk Problems. Pergamon Press, New York (1959)
59. Abdel-Hamid, A.T., Tahar, S., Aboulhamid, E.M.: A public-key watermarking technique for IP designs. In: Design, Automation and Test in Europe, pp. 330–335 (2005)
60. Cui, A.J., Chang, C.H., Tahar, S., Abdel-Hamid, A.T.: A robust FSM watermarking scheme for IP protection of sequential circuit design. IEEE Trans. CAD **30**(5), 678–690 (2011)
61. Kirovski, D., Potkonjak, M.: Intellectual property protection using watermarking partial scan chains for sequential logic test generation. In: High Level Design, Test Verification (1998)
62. Cui, A.J., Chang, C.H.: Intellectual property authentication by watermarking scan chain in design-for-testability flow. In: IEEE International Symposium on Circuits and Systems, pp. 2645–2648 (2008)
63. Chang, C.H., Cui, A.J.: Synthesis-for-testability watermarking for field authentication of VLSI intellectual property. IEEE Trans. Circuits I **57**(7), 1618–1630 (2010)
64. Ziener, D., Baueregger, F., Teich, J.: Multiplexing methods for power watermarking. In: Hardware-Oriented Security and Trust (HOST), pp. 36–41 (2010)
65. Caldwell, A.E., Hyun-Jin, C., Kahng, A.B., Mantik, S., Potkonjak, M., Gang, Q., Wong, J.L.: Effective iterative techniques for fingerprinting design IP. In: Design Automation Conference, pp. 843–848 (1999)
66. Caldwell, A.E., Choi, H.J., Kahng, A.B., Mantik, S., Potkonjak, M., Qu, G., Wong, J.L.: Effective iterative techniques for fingerprinting design IP. IEEE Trans. CAD **23**(2), 208–215 (2004)

67. Lach, J., Mangione-Smith, W.H., Potkonjak, M.: Fingerprinting techniques for field-programmable gate array intellectual property protection. IEEE Trans. CAD **20**(10), 1253–1261 (2001)
68. Chang, C.H., Zhang, L.: A blind dynamic fingerprinting technique for sequential circuit intellectual property protection. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **33**(1), 76–89 (2014)
69. Agrawal, V., Cheng, K.-T.: Finite state machine synthesis with embedded test function. J. Electron. Test. **1**(3), 221–228 (1990)
70. Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptography, pp. 199–203 (1982)
71. NIST Secure Hash Standard (SHS) (2012) http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf
72. Lofstrom, K.: System for providing an integrated circuit with a unique identification. US Patent No. 6,161,213. 12 (2000)
73. Koushanfar, F., Potkonjak, M.: CAD-based security, cryptography, and digital rights management. In: ACM/IEEE Design Automation Conference, pp. 268–269 (2007)
74. Alkabani, Y.M., Koushanfar, F.: Active hardware metering for intellectual property protection and security. In: The 16th Usenix Security Symposium, pp. 291–306 (2007)
75. Alkabani, Y., Koushanfar, F., Potkonjak, M.: Remote activation of ICs for piracy prevention and digital right management. In: IEEE/ACM International Conference on Computer-Aided Design, pp. 674–677 (2007)
76. Potkonjak, M., Nahapetian, A., Nelson, M., Massey, T.: Hardware Trojan horse detection using gate-level characterization. In: ACM/IEEE Design Automation Conference, pp. 688–693 (2009)
77. Sheng, W., Meguerdichian, S., Potkonjak, M.: Gate-level characterization: foundations and hardware security applications. In: Design Automation Conference, pp. 222–227 (2010)
78. Wei, S., Potkonjak, M.: Integrated circuit security techniques using variable supply voltage. In: Design Automation Conference, pp. 248–253 (2011)
79. Wei, S., Koushanfar, F., Potkonjak, M.: Integrated circuit digital rights management techniques using physical level characterization. In: ACM Workshop on Digital Rights Management, Chicago, IL, USA, pp. 3–14 (2011)
80. Wei, S., Nahapetian, A., Nelson, M., Koushanfar, F., Potkonjak, M.: Gate characterization using singular value decomposition: foundations and applications. IEEE Trans. Inf. Forensics Secur. **7**(2), 765–773 (2012)
81. Wei, S., Wendt, J.B., Nahapetian, A., Potkonjak, M.: Reverse engineering and prevention techniques for physical unclonable functions using side channels. In: Design Automation Conference, pp. 1–6 (2014)
82. Zheng, J.X., Li, D., Potkonjak, M.: A secure and unclonable embedded system using instruction-level PUF authentication. In: International Conference on Field Programmable Logic and Applications, pp. 1–4 (2014)
83. Zheng, J.X., Potkonjak, M.: A digital PUF-based IP protection architecture for network embedded systems. In: ACM/IEEE Symposium on Architectures for Networking and Communications Systems, Los Angeles, CA, USA, pp. 1–2 (2014)
84. Zheng, J.X., Potkonjak, M.: Securing netlist-level FPGA design through exploiting process variation and degradation. In: ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, CA, USA, pp. 129–139 (2012)
85. Zheng, J.X., Chen, E., Potkonjak, M.: A benign hardware Trojan on FPGA-based embedded systems. In: International Conference on Field Programmable Logic and Applications, pp. 464–470 (2012)
86. Rajendran, J., Sam, M., Sinanoglu, O., Karri, R.: Security analysis of integrated circuit camouflaging. In: ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, pp. 709–720 (2013)
87. Wendt, J.B., Koushanfar, F., Potkonjak, M.: Techniques for foundry identification. In: Design Automation Conference, pp. 1–6 (2014)

88. Mosteller, F.: Fifty Challenging Problems in Probability with Solutions. Courier Dover Publications (1987)
89. Wei, S., Nahapetian, A., Potkonjak, M.: Robust passive hardware metering. In: IEEE/ACM International Conference on Computer-Aided Design, pp. 802–809 (2011)
90. Sheng, W., Nahapetian, A., Potkonjak, M.: Quantitative intellectual property protection using physical-level characterization. IEEE Trans. Inf. Forensics Secur. **8**(11), 1722–1730 (2013)
91. Dabiri, F., Potkonjak, M.: Hardware aging-based software metering. In: Design, Automation & Test in Europe Conference & Exhibition, pp. 460–465 (2009)
92. Potkonjak, M.: Usage metering based upon hardware aging. US Patent 8,260,708 (2012)
93. Potkonjak, M.: Synthesis of trustable ICs using untrusted CAD tools. In: Design Automation Conference, pp. 633–634 (2010)
94. Koushanfar, F., Fazzari, S., McCants, C., Bryson, W., Sale, M., Song, P.L., Potkonjak, M.: Can EDA combat the rise of electronic counterfeiting? In: Design Automation Conference, pp. 133–138 (2012)
95. Collberg, C., Thomborson, C.: Software watermarking: models and dynamic embeddings. In: ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, TX, USA, pp. 311–324 (1999)
96. Palsberg, J., Krishnaswamy, S., Kwon, M., Ma, D., Shao, Q., Zhang, Y.: Experience with software watermarking. In: Annual Computer Security Applications Conference, pp. 308–316 (2000)
97. Kirovski, D., Drini, M., Potkonjak, M.: Enabling trusted software integrity. In: Architectural Support for Programming Languages and Operating Systems, pp. 108–120 (2002)

IP Protection of FPGA Cores Through a Novel Public/Secret-Key Encryption Mechanism

Bassel Soudan, Wael Adi, and Abdulrahman Hanoun

Abstract Protecting the rights of Intellectual Property (IP) owners is extremely important to the expansion of the core-based design market. Currently, IP providers have no mechanism to guarantee the protection of their IP against over-deployment. We propose a system to guarantee that IP cores can only be deployed into devices agreed upon between the IP provider and the customer. The system is based on secured handshaking with encrypted device and design authentication information. It consists of hardware-supported design encryption and secured authentication protocols. It uses a combination of secret and public-key cryptographic functions devised for an uncomplicated trustable design exchange scenario. The public-key functions use modular squaring (Rabin Lock) on the FPGA chip instead of exponentiation to reduce the hardware complexity. The system limits the parties involved in the transaction to the IP provider and the customer only.

1 Introduction

In core-based design, a customer licenses readily designed Intellectual Property (IP) cores from expert providers and integrates them into their overall system. This allows the in-house design team to concentrate on the differentiating features of the system instead of re-inventing standardized features such as bus interface logic, DSP engines, etc. This is most applicable to FPGA-bound designs where the nature of the FPGA device lends itself to the core-based design style.

While the FPGA market has grown rapidly, the IP core market has failed to achieve its expected growth. The prospect of license violation and IP core

B. Soudan (✉)
Electrical and Computer Engineering Department, University of Sharjah, Sharjah, United Arab Emirates
e-mail: bsoudan@sharjah.ac.ae

W. Adi
Technical University of Braunschweig, Braunschweig, Germany
e-mail: w.adi@tu-bs.de

A. Hanoun
Alcon Laboratories, Inc., Erlangen, Germany

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_11

over-utilization has locked small design-only houses out of the IP providers' market. Currently, IP cores are provided mostly by major device manufacturers who see this as a means for simplifying the development of designs for their devices and therefore increasing demand for the devices themselves. Since most of their profit comes from the sale of the devices rather than the IP cores, any loss due to over-deployment is more than compensated through the increased device sales.

This chapter discusses a secure mechanism and licensing model for protecting against IP over-deployment. The licensing model allows the IP to be licensed for specific uniquely identifiable devices. It is impossible to deploy the IP into any additional device without the explicit involvement of the IP provider (or provable illegal collaboration of the device manufacturer). An IP provider will have absolute control over the number of deployments of the licensed IP. Therefore, the IP provider can charge reasonable per-instance fees and reliably collect all licensing royalties.

In the remainder of this chapter we will first survey existing over-deployment prevention mechanisms; then, the proposed system will be discussed in light of the shortcomings of existing solutions; after that, we will go through certain implementation details for the required hardware module before summarizing and concluding with a brief security analysis of the proposed system.

2 Current Licensing Models

2.1 The Business Case for IP Licensing

Currently, when a customer licenses an IP core from a provider, the customer receives an un-protected, editable, modifiable, and infinitely reproducible net-list to be incorporated into the customer's design. There is nothing to stop an unscrupulous customer from either reverse engineering the net-list to create a locally modified version of the IP; or generating many more instances of the licensed IP core than agreed to in the license agreement. This has led to an expensive business model where IP providers charge large upfront fees to account for run-on possibilities, in addition to the per-instance fee. This fee structure might be agreeable to customers with huge installation bases, but it is prohibitively costly for customers with limited profit margins, the natural customer base for licensing ready-made IP. Since the IP providers cannot limit the number of deployment instances, they cannot differentiate in pricing between the two classes of customers [1]. The main issue is the lack of a reliable mechanism for protecting the intellectual property rights (IPR) of IP providers.

2.2 *Models of IP Over-Deployment and Counter-Measures*

Once an IP leaves the control of the provider, it becomes the target to one of three possible forms of attack: interception during transfer to the customer (or device), duplication by cloners after the IP has been deployed into the market, and possible over-deployment by the customer or the customer's out-sourced device programmer.

Several mechanisms for protecting the FPGA-bound IP have been implemented by manufacturers or proposed by researchers. The following subsections provide an appraisal of the most common IPR protection mechanisms proposed or currently in use.

2.2.1 Non-SRAM Configuration Cells

In its ProASIC^{PLUS} and ProASIC3 lines of FPGAs, Actel replaced volatile SRAM on-chip configuration cells with non-volatile floating gate Flash ROM elements (FROM) [2]. Since FROM elements are non-volatile, the on-board bit stream to allow autonomous re-configuration after power interruption is eliminated. Therefore, there is no bit stream to intercept or clone in the deployed product. The downside is the increased device manufacturing costs.

2.2.2 Encryption of Configuration Patterns

SRAM based FPGAs utilize a volatile configuration memory and therefore require reconfiguration after every power interruption from an external non-volatile memory. The data link between the FPGA and the non-volatile memory represents a significant security risk. To help protect against interception and possible cloning, major FPGA device manufacturers have enabled the encryption of the design data to ensure that it cannot be meaningfully intercepted between the non-volatile memory and the device [3, 4].

Actel supported the use of encrypted design bit streams by integrating an AES block cipher core into the device. The AES block is used to decrypt the design bit stream as it is being loaded into the device. The decryption key is programmed into the target device typically at the IP Provider's own facility. The decryption key is stored in secure on-chip FROM. After the key has been programmed into the device, it only accepts a design bit stream that has been encrypted with the same key. Xilinx on the other hand, integrated DES and Triple DES decryptors into their devices [4,5]. In both solutions from Actel and Xilinx intercepting the design bit stream is useless and cloning is impossible without knowing the exact decryption key.

The decryption keys need to be programmed into the devices. This presents three scenarios:

- The IP provider and the device manufacturer are the same entity. The device manufacturer initializes the devices with the keys (and possibly downloads the licensed IP into the devices) before sending them to the customer. However, our aim is to open the IP core market to allow small-scale IP providers to participate. The licensing model should not be built on the premise that the IP provider and device manufacturer are the same entity.

- The IP provider will program the keys into the devices. For this to work, the provider must have physical possession of the devices. But, the licensing model should not require shipping the devices to the IP provider for key programming and then shipping them back to the customer for programming. It is easily conceivable that the device manufacturer, IP provider and IP customer are in very distant geographical locations. IP provider should not need to have any contact with the devices; even for key programming.
- The decryption keys may be shared with the device manufacturer for initialization into the devices. However, these keys must then also be shared with the customer because it is the customer who will eventually need to encrypt the design bitstream before storing it in the configuration memory. The end result is that the customer will have the undesired access to the raw description of the licensed IP.

2.2.3 Detecting IPR Violation

There have been several proposed methods for detecting when an IPR violation occurs. Most are based on fingerprinting the IP or embedding hidden watermarks in the design description [6–12]. While these methods may be successful in determining when an IPR violation has occurred, they are not useful in preventing the violation in the first place, which is our aim in this work. One needs to suspect that a violation has occurred and have a clue of where it has occurred before being able to detect it. The sheer number of FPGA based designs continuously appearing on the market makes it prohibitive to even contemplate checking every single design for possible IPR violations.

IP providers have resorted to overt and covert customer audits in order to detect IPR violations. Regular (covert) monitoring of the customer's in-market products ensures that the licensed IP has not made it surreptitiously into products not covered by the license agreements. In addition, overt on-site audits let the customers know that the IP provider is scrutinizing the adherence. These methods impose an extra financial burden on the IP provider. In addition, both forms evoke a sense of lack of confidence in the customer and are counter to the theories of business based on trust. This is probably one of the main reasons why the IP market remains very small and concentrated amongst device manufacturers who derive their revenue mainly from the devices and not the IP royalties. To them providing IP for licensure is primarily a means to boost device sales.

2.2.4 Pay-Per-Use Licensing

A pay-per-use methodology based on secret-key cryptography has been proposed to prevent the over-deployment of licensed IP [1, 13]. The methodology utilizes an e-commerce server at a *Trusted-External-Party* (TEP) to regulate all transactions related to IP licensing. The TEP server builds a device database about *all* devices from *all* device manufacturers. In addition, the TEP server maintains an IP core database based on information supplied by the IP core providers *for each licensable* IP core. Finally, the TEP server also needs to maintain a database of *all* IP customers and the IPs they have licensed.

Upon purchasing an IP core, the customer uses *trusted software* that communicates with the TEP server to download an encrypted copy of the licensed IP, decrypt and then re-encrypt it for programming into each particular device. The TEP server also handles charging the customer for the instances of the core being downloaded to the devices [1]. When the encrypted bit stream is downloaded into the device, the device's secret identifier is used in conjunction with data pre-pended to the design bit stream to obtain the decryption key needed to decrypt the rest of the bit stream [1]. Figure 1 shows the business relationships required for this proposal to work properly.

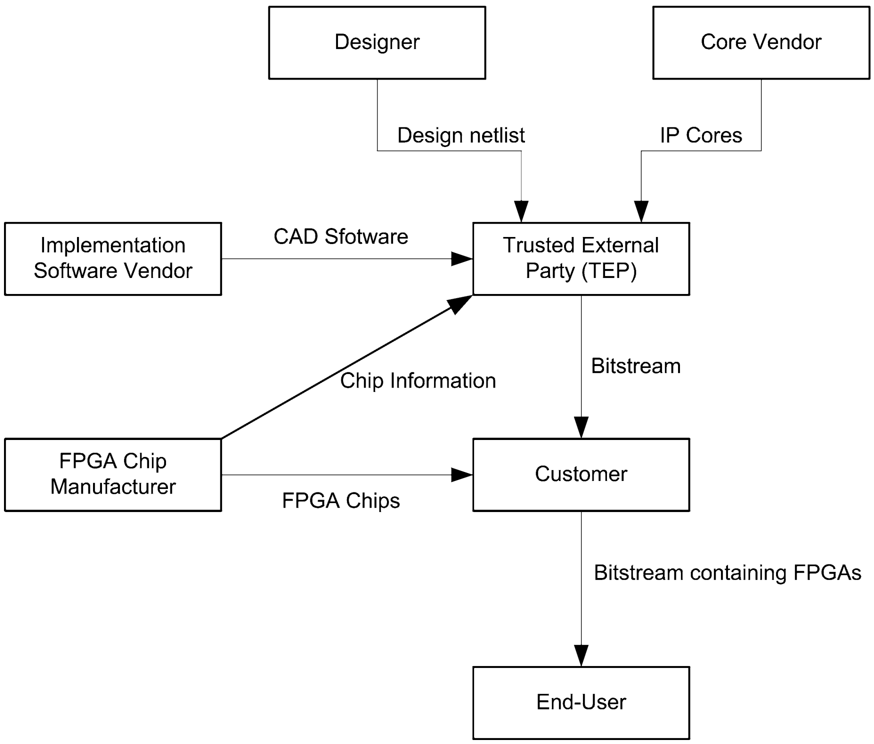


Fig. 1 Business relationships amongst parties involved in designing and using FPGAs [13]

While the methodology proposed above seems to address the issue of preventing over-deployment, it suffers from several drawbacks:

- The whole methodology is centered on the TEP server which now presents a single point of complete failure for the system. Should the TEP server fail or be compromised, the whole system fails.
- The TEP server is entrusted with a lot of sensitive information about the chips, the cores, the device manufacturers, the IP providers, and the customers. The TEP is also expected to handle payments of customers to IP providers. The TEP is expected to maintain data on every device from every device manufacturer and every core from every IP provider. The device data needs to be kept permanently to allow for later device re-configuration. There is a strong possibility of data explosion at the TEP.
- The TEP server downloads an encrypted design bit stream to the customer. This bit stream is decrypted at the customer's site using the trusted programming software and is then re-encrypted with a token received from the TEP. This means that the raw decrypted design bit stream will exist for some instance of time on the customer's premises exposing it to possible compromise.

While the pay-per-use licensing model sounds reasonable theoretically, widespread implementation of such a system poses many practical issues.

2.2.5 System Based on Secured-Handshaking

In this chapter we propose a thorough business model which gives the IP provider exact control over the number of deployments for the licensed IP. The proposed business model includes minor changes in the design of the FPGA itself, the design flow, and the IP exchange protocol. It may seem that the proposal being set forth in this work will complicate the IP core licensing mechanism. However, given that IP core providers will have exact control over the number of deployments, the business model can shift into the more economically sound supplier/client model instead of the semi-partnership that it is today. IP providers will not need to charge the large basic rates to protect against "what-if" scenarios. They can charge on a per instance basis as they will have exact control over how many instances are being produced. It is expected that the average per-instance cost for IP clients will be lower compared to the current model. This should entice more clients into the market and should greatly expand the market for licensed IP cores.

3 Proposed Scheme

As it was mentioned earlier, the main issue with the current IP licensing model is the lack of a reliable mechanism for the IP providers to reliably extract revenue from the deployment of their licensed IP. The issue is in the nature of the IP description provided to the customer. IP providers currently communicate a raw description of the IP to allow the IP customer to integrate it into the finished product. With the customer having access to the raw description, it becomes essentially impossible to protect the licensed IP against over-deployment, reverse engineering, or partial paraphrasing. Therefore, the solution seems to be in denying the customer access to the raw description of the licensed IP under any condition. This is analogous to preventing the licensee of a software product from accessing the source code of the product. Except that, we also need to prevent the customer from being able to duplicate the compiled product as well.

While software IP providers cannot prevent the customer having duplication access to the compiled product, most have been able to protect their interest by requiring a license every time the product is invoked. This license can be in one of several forms: a license file, an access key based on some user-specific identifier, or hardware specific identifiers. The wide availability of counterfeit or cracked software products are a strong evidence of the failure of such mechanisms in preventing software over deployment. The one exception is the hardware specific identifier which has met with reasonable success compared to the other systems.

3.1 Hardware Specific IP Licensing

A licensing model based on hardware specific identifiers typically depends on uniquely-identifying hardware features such as a processor ID, a permanent serial number, or some other customer inaccessible identifier. Should the customer want to license the IP for multiple devices, that many device-specific licenses would be generated and used individually on the matching devices. From the IP provider's perspective, this methodology provides reasonable protection against unlicensed use of the IP. Most software IP providers employing this licensing technique automate the process through an automatic response system on their website. The customer supplies a purchase order number—which indicates the number of approved license instances—and a machine's uniquely-identifying feature. The automated server ensures that the maximum number of instances has not been exceeded and generates the appropriate license and provides it to the customer in one of many physical means.

Provided that a uniquely-identifying device feature can be established, the "per-device" licensing concept can be extended to the licensing of hardware IP for use in FGPA devices. The main difference is the lack of license files in hardware IP licensing. The exchange must be based on multiple device-specific IP descriptions. In addition, the IP descriptions must be in a user-unreadable form to prevent reverse engineering or illegal paraphrasing of the licensed IP into a customer's own design.

3.2 The Proposed Business Model

Our proposed licensing model is built around the premise that the provider will communicate to the customer device-specific versions of the IP description encrypted using device-specific keys inaccessible by the customer. The model requires the creation of two identical copies of the encryption/decryption secret key at the IP provider's site and securely on the device at the IP customer's site. To avoid issues related to key management, the identical pairs of secret keys will be created through the exchange of open public values coupled with a challenge-response system. Using the encryption key, the IP provider creates a distinct encrypted copy of the IP description per device. The encrypted IP descriptions are communicated to the IP customer. The customer downloads each encrypted copy of the IP into the appropriately matching device. The encrypted description will be decrypted on the device itself using the key that was created earlier.

The most favorable features of this licensing model are the very limited involvement of the IP provider and hardware manufacturer and the exact limits on the number of licensed IP instances. When the "per-device" model is combined with IPR Protection mechanisms through encryption, the resulting licensing model affords the IP providers and IP customers with all the necessary features needed for a trustful financially-sound business transaction.

The exact details of the exchange will be put forward in the next subsections.

3.3 Business Model Enablers

The system is designed to satisfy the following requirements:

- The IP provider must be able to exactly limit the number of IP instances and the manner of their usage. Therefore, the raw IP description must not exist outside the IP provider's control at any point in time. The one exception is on the device itself under the condition that the design bit stream must not be extractable from the device after programming.
- The licensing model should be based on the generic situation of an IP customer licensing an IP from an IP provider and a surrogate programming facility handling the actual programming of the devices. Therefore, the IP provider should not need to have direct interaction with the devices.
- The IP should be securely distributable over an open channel like the Internet without the need for sharing of secrets such as encryption/decryption keys between IP provider and customer or surrogate programmer. The system security should be based on known unbroken ciphering technology.
- The system response time and hardware complexity must be kept as low as possible.
- Collaboration between device manufacturer and IP customer should not lead to breaking the system.

A hardware mechanism for generating the secret keys on the device and a specific exchange protocol are needed to fulfill these requirements.

3.3.1 The Hardware Module

To enable the creation of the unique device-specific key pair, a Device Identity Module (DIM) like the one shown in Fig. 2 needs to be built into each device. The DIM should reside in a tamper-proof area where no attack would be possible under any operating mode.

For each device, the manufacturer establishes a unique public *device identity* (DI), which can be branded on the device itself and/or stored in a readable area in the device.

The DIM needs to include the following elements:

- A non-volatile unreadable storage location to hold a *secret device identity* (*SDI*). The SDI is mapped from DI such that no key collision is possible:

$$\mathrm{SDI} = \mathrm{F}(\mathrm{DI},\ \mathrm{SMK}) \tag{1}$$

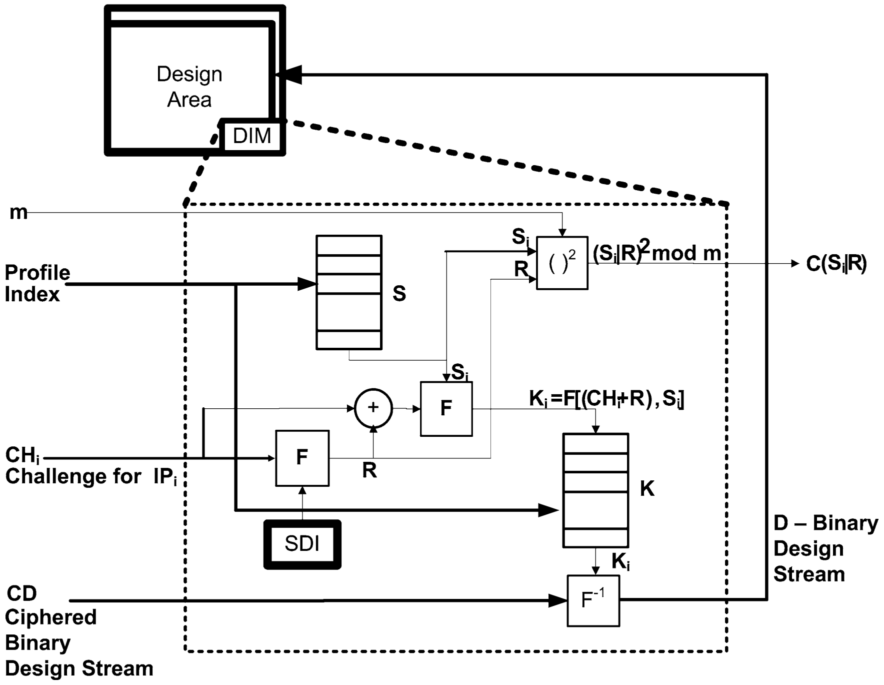


Fig. 2 Architecture of a possible FPGA Device Identity Module (DIM)

where SMK is the manufacturer's Secret Master Key for that particular FPGA type and F is a strong block cipher. The manufacturer should embed the SDI value before delivering any device and be responsible for the uniqueness of all SDI's and DI's. The manufacturer is also responsible to keep SMK secret.

- A write-once register file *S* whose contents should be fully random and not readable by the IP customer. The register file is indexed through a profile index which will be provided from outside the chip.
- A modular squaring block *()*^{2} *modulo m*, where m is a 500–1000 bit input which will be obtained from the IP provider [14]. The input m does not need to be stored locally in the DIM.
- A register file *K* to hold the decryption keys. It must be inaccessible from outside the device under any condition. The register file K is indexed through the same profile index as the register file S.
- A hardware decipher block F^{-1}.

The function F (and its inverse F^{-1}) should be a strong cipher such as AES [15]. The size of all registers, secret keys and other vectors should be 128 bits, the same as the cipher block size. Whenever the mapping $Y = F(X, K)$ is used, it means that X corresponds to the clear text, K to the key and Y to the resulting encrypted text.

3.3.2 The Exchange Protocol

After the IP customer has selected the IP to license from the IP provider, a secured design transfer protocol has to be followed to enable the licensing model. The protocol can be expressed as follows (refer to the 4-step protocol in Fig. 3):

1. The IP provider publishes a random challenge CH_i, which should change frequently with a time stamp defining its lifetime. The IP provider also selects two secret prime numbers p and q, and publishes their product m. The primes p and q should be between 500 and 1000 bits to ensure a high security level. CH_i and m can be publicly advertised on the IP provider's website without loss of security.
2. The IP customer obtains the challenge CH_i and m from the IP provider's web page, applies the IP provider's profile ID, the challenge CH_i and the value m into each of the target devices. This operation freezes the value of the ith entry in the S register file (S_i) in each device and causes the device to produce the ciphered concatenation CR_i as shown in expression (2) below.

$$CR_i = \left(S_i \middle| R\right)^2 \mod m \tag{2}$$

where R is calculated inside the DIM based on CH_i and SDI as shown in expression (3) below.

$$R = F(CH_i,\ SDI) \tag{3}$$

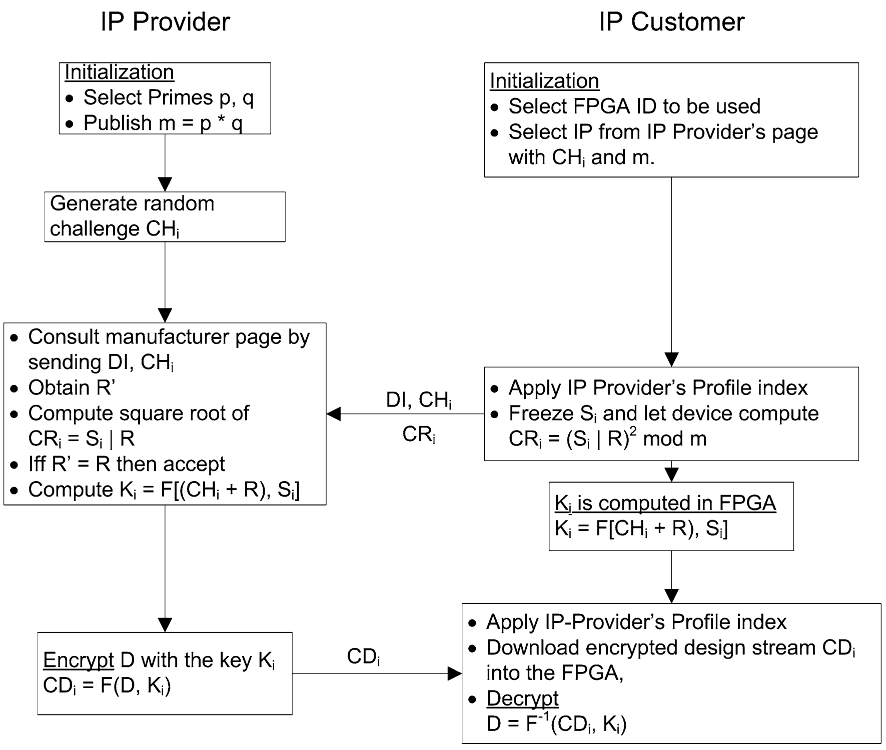

IP Provider
IP Customer
Initialization
• Select Primes p, q
• Publish m = p * q
Generate random challenge CHi
• Consult manufacturer page by sending DI, CHi
• Obtain R'
• Compute square root of CRi = Si | R
• Iff R' = R then accept
• Compute Ki = F[(CHi + R), Si]
Encrypt D with the key Ki
CDi = F(D, Ki)
Initialization
• Select FPGA ID to be used
• Select IP from IP Provider's page with CHi and m.
• Apply IP Provider's Profile index
• Freeze Si and let device compute CRi = (Si | R)2 mod m
Ki is computed in FPGA
Ki = F[CHi + R), Si]
• Apply IP-Provider's Profile index
• Download encrypted design stream CDi into the FPGA,
• Decrypt D = F-1(CDi, Ki)
DI, CHi
CRi
CDi

Fig. 3 Secured design transfer protocol

In addition, the operation causes a secret key K_i unique for each IP provider and target device to be generated inside each device according to expression (4) below.

$$K_i = F\ [(CH_i + R)\,,\ S_i] \tag{4}$$

The customer collects the device identity DI and CR_i for each device and sends them to the IP provider along with the purchase order.

3. The IP provider uses a public challenge-response verification engine (VE) on the device manufacturer's web page to generate a response R' in a manner similar to expression (3) above. The verification engine generates R' based on the IP provider's CH_i and the device identity DI received from the IP customer as shown in Fig. 4. R' will be exactly identical to R.

$$R' = F\,(CH_i,\ F\,(DI,\ SMK)) \tag{5}$$

The IP provider uses p and q to compute the square roots of CR_i. The number of square roots over the ring Z_m is 4. The correct root is the one that corresponds

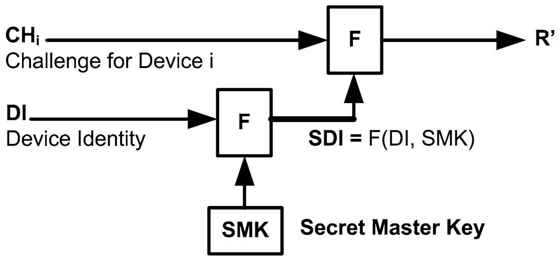


Fig. 4 "Verification Engine" published by the manufacturer to verify device identities

to the concatenation ($S_i|R$), it is the root with R' in the least significant bits. The IP provider can extract S_i from the concatenation and generate a matching K_i for each device following the operation in expression (4). *The IP provider is the only participant in the entire exchange that knows the exact value of S_i and therefore K_i.*

4. The IP provider uses the device-unique K_i to encrypt the Binary Design Stream (D) and produce a unique CD_i for each device as shown in expression (6).

$$CD_i = F\ (D,\ K_i) \tag{6}$$

The resulting ciphered binary design streams CD_i are sent to the customer.

5. The customer applies the IP provider's profile ID and downloads each CD_i to the device with the matching DI. CD_i will be automatically decrypted as the right K_i has already been generated in the device in step 2 earlier.

$$D = F^{-1}\ (CD_i,\ K_i) \tag{7}$$

The above protocol can run in parallel for all required devices. All of the steps of the protocol can be quite easily automated to simplify and speed up the operation.

3.4 Integrating Multiple IPs

For the proposed protocol to be practically applicable it must not overly restrict the IP licensing business model. Handling a design based on a single licensed IP is trivial. Through the protocol described above, the IP provider supplies encrypted bitstreams for each of the target devices; the IP customer (or the contract programming facility) downloads the bitstreams into the devices and the transaction is concluded. Of more interest is the scenario where the IP customer intends to integrate multiple IPs licensed from multiple IP providers along with in-house developed logic [16].

One of the main requirements stated earlier was that the IP customer must never have possession of the IP's raw description to guarantee absolute IPR protection. Therefore, the integration of the licensed IP within the design flow must be performed in a black-box manner.

3.4.1 Black Box Simulation and Analysis

Of paramount interest during the design phase is the ability to simulate and analyze the design before committing to hardware. To enable overall design simulation at the IP customer's site, The IP provider shares compiled simulation and timing models for the licensed IPs [17–21]. These compiled models protect the details of the licensed IP but still allow the customer to integrate the IP into the functional simulation and timing analysis of the overall design. Compiled models are supported by most currently available design environments and typically are accurate enough to mimic the inclusion of the actual IP in the overall design. In addition, the compiled models can also be fully portable across a variety of design and simulation environments.

3.4.2 The Physical Integration of the IPs

The physical integration of the licensed IPs has to occur on the device itself. The bitstreams of the different licensed IPs need to be downloaded to the device separately because they are encrypted with different keys. To enable the final physical integration of the licensed IPs with the overall design, the technique of partial reconfiguration will be utilized [22–29]. Partial reconfiguration allows a portion of the chip to be reconfigured whilst the remaining parts of the chip retain their current configurations [30].

The design of older devices (like the Virtex 2 from Xilinx) constrained partial reconfiguration to blocks that consist of integer multiples of Configurable Logic Block (CLB) columns spanning the entire height of the chip [31–34]. The design of modern devices has eliminated some of these limitations. The Virtex 4 and Virtex 5 devices are based on a tiled design [35]. The CLB array is arranged into configuration rows that span only 16 CLBs vertically [36]. Each frame spans the vertical height of a tile rather than the full height of the device. This new tiled design will allow blocks to be of any shape with the single condition that vertically they must span multiples of 16 CLBs.

This restriction is reasonably fair. Actually, the heterogeneous nature of modern FPGAs, in itself, places similar floorplanning restrictions on most designs [37]. Newer FPGA chips consist of columns of CLBs, with column pairs of RAMs and multipliers interleaved between them. The heterogeneous logic and routing resources on a modern FPGA force strict requirements on the floorplanning of blocks based on the utilization of resources.

3.4.3 High Level Design Flow for the Integration of Multiple Licensed IPs

The following is a description of the overall design flow for integrating multiple licensed IPs from licensing the IP's all the way through device programming [38]:

1. The IP providers advertise their IPs including resource requirements (number of CLBs and IOBs) for different device types.

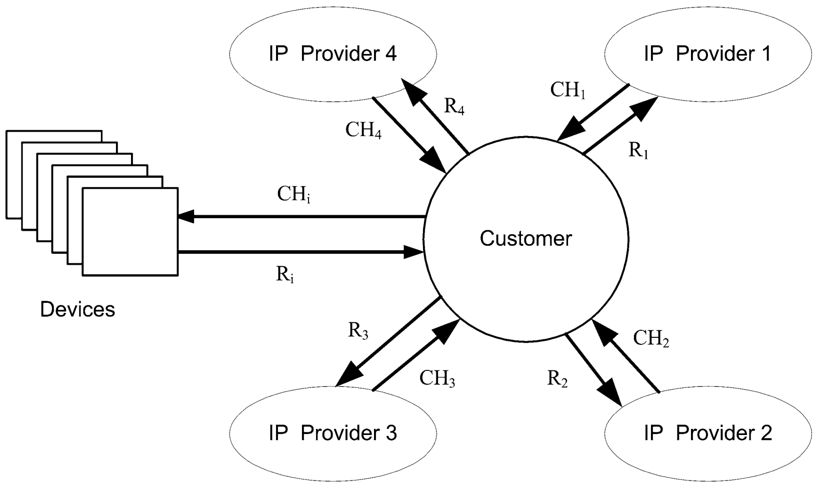


Fig. 5 Interaction between IP providers, customer and devices to create profiles and decryption keys on the devices

2. The IP customer selects the IPs to be licensed and creates a floorplan where each licensed IP conforms to the reconfiguration limits imposed by the chosen device type. The area allocated for each IP must contain at least as many CLBs as required by the IP provider.
3. Upon establishing the purchase agreement with the IP providers, the customer creates profiles on each device, one per IP provider. The profiles are created based on the challenges CH_i from the IP providers as discussed earlier. This process also creates the secret keys for the individual IP providers inside the devices (as shown in Fig. 5).
4. As part of the purchase agreement, the IP customer also communicates to the IP provider the floorplan requirements for the licensed IP for the given device types. The IP customer communicates the challenge response (R from expression (3) shown earlier) to the IP provider. The IP provider uses R to generate a duplicate copy of the encryption key using expression (4).
5. The IP provider supplies compiled simulation and timing analysis models of the licensed IP to the customer. The IP provider generates a design bitstream for the IP that meets the floorplan requirements, encrypts it and dispatches it to the IP customer. The involvement of the IP provider is complete at this point. The IP customer collects the encrypted design bitstreams for all licensed IPs from the different IP providers as shown in Fig. 6.
6. Using "device management" software, the customer builds an association between each device and the specific design bitstream(s) from each IP provider. Using this association and the floorplan generated earlier, the software prepends

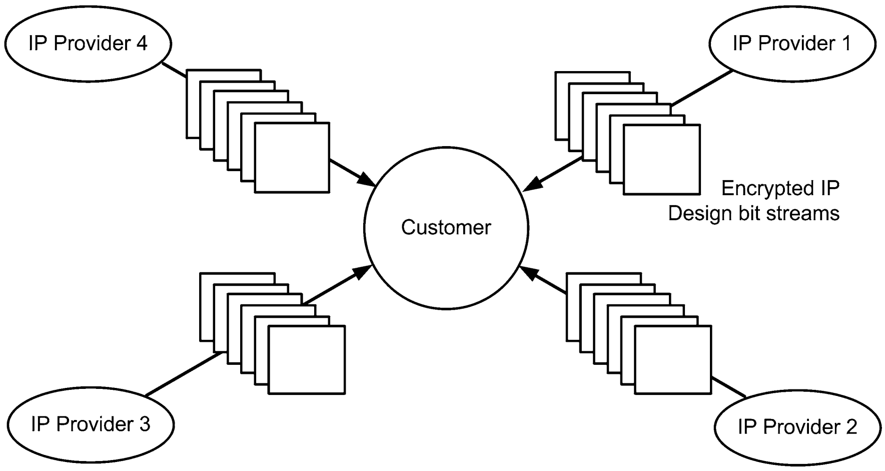


Fig. 6 The customer collects encrypted IP design bit streams from each IP provider for each device

to each encrypted design bitstream an unencrypted header that specifies the location on the device where the IP macro is supposed to be programmed. The software also creates a device programming script for each device as shown in Fig. 7. This script issues appropriate JTAG commands understandable by the DIM to switch to the appropriate profile before the bitstream of each IP is downloaded. Switching to the profile matching the specific IP allows the proper secret key (K_i) to decrypt the incoming encrypted bitstream.

7. As the script is executed, the bitstreams are downloaded into the device. The DIM recognizes the unencrypted location information in the header and passes it along to the rest of the chip. As the encrypted IP description passes through, it gets decrypted and programmed in the appropriate location on the chip.

3.5 Automating the Process

A few simple software segments can automate the process and remove a great deal of the interaction complexity.

3.5.1 Device Management Software

A design consisting of several licensed IPs and several in-house designed macros to be programmed into a large number of devices can result into a very large number of individual files to be managed and organized. Device management software can

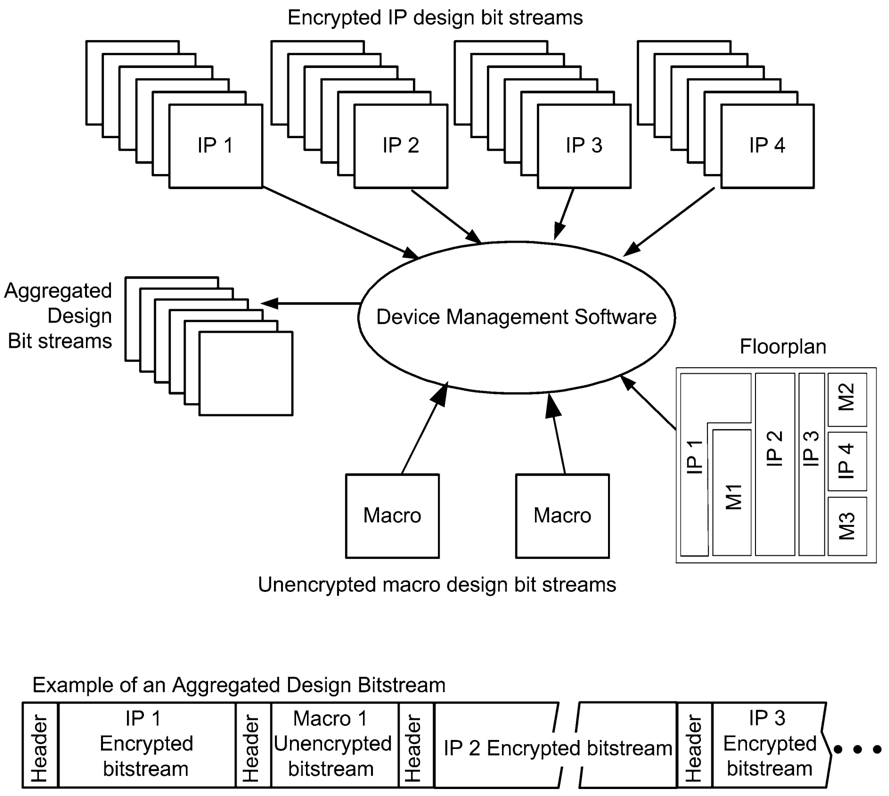

Encrypted IP design bit streams
IP 1
IP 2
IP 3
IP 4
Aggregated Design Bit streams
Device Management Software
Floorplan
IP 1
M1
IP 2
IP 3
M2
IP 4
M3
Macro
Macro
Unencrypted macro design bit streams
Example of an Aggregated Design Bitstream
Header
IP 1 Encrypted bitstream
Header
Macro 1 Unencrypted bitstream
Header
IP 2 Encrypted bitstream
Header
IP 3 Encrypted bitstream

Fig. 7 The device management software produces device programming scripts for the different devices

be easily developed to handle the following tasks and simplify the administration of the process:

1. Identify which IP goes where on the device through a floorplan that it receives from the design software including the profile ID for each IP.
2. Manage the encrypted bitstream files for each IP for each device. The software identifies the device of each IP file, possibly from a simple header enclosed by the IP provider.
3. Creates a programming script for each device. The script contains the names of the IP files for the specific device and the JTAG commands to be issued before downloading each IP file as well as location data for each IP taken from the floorplan.
4. The device owner's designs are to be treated like any other IP. If the device owner elects not to encrypt the description for in-house developed macros, the software will indicate that a special "owner" profile should be used during the download process.

3.5.2 Device Programming Software

The device programming software needs to be modified slightly to handle additional tasks. The required modifications depend on the method for downloading the IP bitstreams to the devices. Two possible methods exist: individual bitstreams and a combined bitstream (Fig. 8).

One possibility is for the bitstreams for the IPs to be maintained separately and downloaded into the device individually under the control of the programming script generated by the Device Management Software. In this case, the device management software should prepend the bitstream of each licensed IP with its proper location information. This information does not need to be encrypted. The DIM recognizes the un-encrypted header section and passes it along to the rest of the circuitry. The device programming software needs to send the appropriate profile index to the DIM before the bitstream of each IP is downloaded. The DIM will use the proper key to decrypt the IP's bitstream and send it to the appropriate section of the chip. Before downloading the next IP's bitstream, the profile index is switched to the appropriate profile. A special "owner" profile can be created to allow downloading the unencrypted bitstreams of the owner's in-house developed modules.

The other possibility is to combine all bit streams for the device into a conglomeration. The independently encrypted bitstreams of the licensed IPs as well as the unencrypted bitstreams of the owner's macros are merged into a single bitstream. The different bitstreams will be separated within the merged bitstream by profile identifiers. As the merged bitstream is downloaded into the device, the DIM recognizes the profile identifiers and switches decryption keys as the different bitstream sections pass through it. In this case, partial reconfiguration capabilities may not be required, but block floorplan requirements must still be maintained.

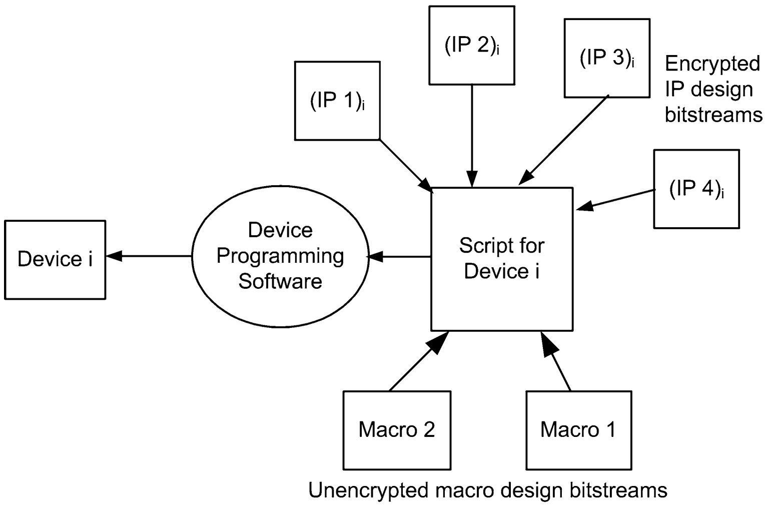

(IP 1)i
(IP 2)i
(IP 3)i
(IP 4)i
Encrypted IP design bitstreams
Device i
Device Programming Software
Script for Device i
Macro 2
Macro 1
Unencrypted macro design bitstreams

Fig. 8 Device programming software uses the programming scripts to download the proper bitstreams to proper devices

4 Analysis

This section discusses the possible security threats and attack points as well as advantages and disadvantages of the proposed system.

4.1 Security Threats

The proposed system is breakable in only two possible scenarios:

1. The system depends on a unique DI/DIM pair and a randomly generated set of S_i's for each device. The only way the system can be broken is if the manufacturer generates devices with duplicate DI/DIM pairs *and* makes the S_i generator act in a deterministic manner. The only party in the whole exchange that gets to know the exact value of S_i for a specific device is the IP provider. Therefore, the IP provider can easily determine and prove such collusion.
2. The device manufacturer may build a backdoor into the device where the customer can access the decrypted BDS directly. This is a highly unlikely scenario as it would be easily provable and would destroy the manufacturer's credibility.

The system security is as good as Rabin Lock which is based on factoring known to be still unbreakable. The fact that S_i is unpredictable to all parties gives the system a special strong security level. Combining secret and public-key systems in a joint security function is an additional argument in favor of its security.

4.2 Advantages

The proposed system has several advantages. The main advantages are:

- IP core providers can exactly control the number of copies being produced of their cores. Given that the CD is encrypted based on a particular device's SDI, the IP provider can be sure that the resulting CD cannot be used to program any other device. Short of cloning the actual devices (including the unique SDI and random S_i's), the customer (or any other party) cannot over-deploy a licensed IP.
- The proposed mechanism allows the customer to pay for the exact number of instances the IP is instantiated. It is expected that the average per-instance cost for IP clients will be lower compared to the current model. This should entice more clients (especially low volume clients) into the market and should greatly expand the market for licensed IP cores.
- Only the IP provider and customer are involved in the exchange. The device manufacturer's involvement is not required beyond the manufacturing of devices. IP providers are not required to have firsthand possession—even temporarily—of

the devices for key programming. The communication of the secured IP is carried out electronically and dealing with the physical devices is left to the IP customer or contract programmer. The involvement of each party is required only within the party's sphere of expertise and interest.

- The proposed system does not preclude advanced functionality such as dynamic reconfiguration. If the block to be loaded into the FPGA during the reconfiguration is a licensed IP, the FPGA must be configured during initialization with an appropriate decryption key from the IP's provider like any other licensed IP. The proper profile index can be specified during the reconfiguration to switch the DIM to use the appropriate decryption key.
- The hardware needed to enable the system is negligible compared to available resources in a modern device. The entire DIM would consume less than 0.1 % of the resources in a modern device like the Xilinx Virtex 4.

4.3 Disadvantages

The main disadvantage of the system is that it reduces flexibility from a contract programmer's point of view. Once devices have been initialized for a specific customer design, they cannot be used for any other design or any other customer. Also, only these specific devices can be used for the particular design.

While the proposed system may seem to complicate the interaction, most of the complexity can be very easily hidden through software automation as described in Sect. 3.5.

5 Conclusion

This chapter discussed a mechanism for the authenticated transfer of design information that prevents IPR violations in an FPGA design environment. The novel security technology uses combined public and secret-key cryptographic mechanisms. The resulting system offers a high level of security while still being reasonably easy to handle. The proposed mechanism allows design distribution over public networks without loss of security. The mechanisms employed are based on trustable cryptographic primitives well known in secret-key and public-key cryptography, but utilizes low complexity functions to simplify the implementation and save resources. In particular, the selected public-key technology employs squaring in a ring resulting in the simplest public key system known to date.

The system appears to be unbreakable even if the device manufacturer collaborates with the IP customer. The design transfer does not involve the manufacturer on-line and does not need the IP provider to have any contact with the FPGA devices. All transactions can run over any open communication network without prior secret sharing. The FPGA manufacturer must however be trustworthy to manufacture according to the specified hardware security architecture and not to have built backdoors in the FPGA architecture.

The hardware resources needed to implement the system are in the range of 0.1 % of total available resources in a modern device similar to a Virtex 4 from Xilinx. An insignificant overhead compared to the level of IPR protection afforded.

References

1. Kean, T.: Cryptographically enforced pay-per-use licensing of FPGA design intellectual property. In: Proceedings International Workshop on IP Based Design (2002)
2. Actel: Implementation of Security in Actel's ProASIC and ProASIC$^{\underline{\text{PLUS}}}$ Flash-Based FPGAs. Application Note AC185, September 2003
3. Actel: ProASIC3/E Security. Application Note. XCell J., 40, p. 29, 10 January 2005
4. Peattie, M.: Use triple DES for ultimate Virtex-II design protection. XCell J., 29 (2001)
5. Tseng, C.W.: Lock your designs with the Virtex-4 security solution. XCell J. (2005)
6. Castillo, E., Meyer-Baese, U., Garcia, A., Parrilla, L., Lloris, A.: IPP@HDL: efficient intellectual property protection scheme for IP cores. IEEE Trans. VLSI Syst. **15**(5), 578–591 (2007)
7. Kirovski, D., Hwang, Y.Y., Potkonjak, M., Cong, J.: Protecting combinational logic synthesis solutions. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **25**(12), 2687–2696 (2006)
8. Lach, J., Mangione-Smith, W., Potkonjak, M.: Fingerprinting techniques for field-programmable gate array intellectual property protection. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **20**(10), 1253–1261 (2001)
9. Torunoglu, I., Charbon, E.: Watermarking-based copyright protection of sequential functions. IEEE J. Solid State Circuits **35**(3), 434–440 (2000)
10. Yuan, L., Pari, P.R., Qu, G.: Soft IP protection: watermarking HDL Codes. In: 6th Information Hiding Workshop, May 2004. Lecture Notes on Computer Science, vol. 3200, pp. 224–238. Springer, Heidelberg (2004)
11. Newbould, R.D., Carothers, J.D., Rodriguez, J.J., Holman, W.T.: A hierarchy of physical design watermarking schemes for intellectual property protection of IC designs. Proc. Int. Symp. Circuits Syst. **IV**, 862–865 (2002)
12. Koushanfar, F., Potkonjak, M.: CAD-based security, cryptography, and digital rights management. In: Proceedings of the IEEE/ACM Design Automation Conference, June 2007, pp. 268–269 (2007)
13. Kean, T.: Method of protecting intellectual property cores on field programmable gate array. US Patent US20020199110 A1, 26 December 2002
14. Adi, W.: Fuzzy modular arithmetic for cryptographic schemes with applications to mobile security. In: Proceedings of the IEEE International Conference European Conference, pp. 263–265 (2000)
15. Federal Information Processing Standards Publication: FIPS 197, AES, Advanced Encryption Standard (2001)
16. Wagner, F., Cesario, W., Carro, L., Jerraya, A.: Strategies for the integration of hardware and software IP components in embedded systems-on-chip. Integr. VLSI J. **37**(4), 223–252 (2004)
17. Bergamaschi, R., Bhattacharya, S., Wagner, R., Fellenz, C., Muhlada, M., Lee, W., White, F., Daveau, J.M.: Automating the design of SOCs using cores. IEEE Des. Test Comput. **18**(5), 32–45 (2001)
18. Biggs, J., Gibbons, A.: Reference methodology for enabling core based design. In: European Synopsys Users Group (2002)

19. Coussy, P., Baganne, A., Martin, E.: A design methodology for IP integration. Proc. IEEE Int. Symp. Circuits Syst. **IV**, IV-711–IV-714 (2002)
20. Gajski, D., Wu, A., Chaiyakul, V., Mori, S., Nukiyama, T., Bricaud, P.: Essential issues for IP reuse. In: Proceedings of the IEEE Design Automation Conference, San Francisco, pp. 37–42, June 2000
21. Wala, M., Bouldin, D.: Integrating and verifying intellectual property blocks using platform express and modelsim. In: Proceedings of the 48th Midwest Symposium on Circuits and Systems, vol. 1, pp. 758–761, 7–10 August 2005
22. Ababei, C., Bazargan, K.: Non-contiguous linear placement for reconfigurable fabrics. In: Proceedings of the 18th International Parallel and Distributed Processing Symposium (2004)
23. Banerjee, S., Bozorgzadeh, E., Dutt, N.D.: Integrating physical constraints in HW-SW partitioning for architectures with partial dynamic reconfiguration. IEEE Trans. Very Large Scale Integr. Syst. **14**(11), 1189–1202 (2006)
24. Blodget, B., McMillan, S., Lysaght, P.: A lightweight approach for embedded reconfiguration of FPGAs. In: Proceedings of the Design, Automation and Test in Europe Conference, pp. 399–400 (2003)
25. Rana, V., Santambrogio, M., Sciuto, D.: Dynamic reconfigurability in embedded system design. In: Proceedings of the International Symposium on Circuits and Systems, pp. 2734–2737 (2007)
26. Chu, A., Miller, S., Sima, M.: Reconfigurable solutions for very-long arithmetic with applications in cryptography. In: Proceedings of the Great Lakes Symposium on Very Large Scale Integration, Orlando, pp. 59–64, 4–6 May 2008
27. Brebner, G.J., Neely, C.E.: Method and system for preparing modularized circuit designs for dynamic partial reconfiguration of programmable logic. US Patent No. 8,560,996, 15 October 2013
28. Xilinx Inc.: Partial Reconfiguration. Development System Reference Guide. Chapter 5, pp. 113–140 (2005)
29. Xilinx Inc.: Partial Reconfiguration User Guide. UG 702, v. 12.3, 5 October 2010
30. Zeineddini, A.S., Gaj, K.: Secure partial reconfiguration of FPGAs. In: Proceedings of the 2005 IEEE International Conference on Field-Programmable Technology, pp. 155–162, 11–14 December 2005
31. Gericota, M., Alves, G., Silva, M., Ferreira, J.: Run-time management of logic resources on reconfigurable systems. In: Proceedings of Conference on Design and Test in Europe (2003)
32. Mesquita, D., Moraes, F., Palma, J., Möller, L., Calazans, N.: Remote and partial reconfiguration of FPGAs: tools and trends. In: Proceedings of the International Parallel and Distributed Processing Symposium (2003)
33. Xilinx: Difference-Based Partial Reconfiguration. Application Note Number 290, v. 2.0, 3 December 2007
34. Xilinx Inc.: Virtex Series Configuration Architecture User Guide. Application Note Number 151, v. 1.7, 20 October 2004
35. Xilinx Inc.: Virtex-4 FPGA Configuration User Guide. UG 071, v1.11, 9 June 2009
36. Sedcole, P., Blodget, B., Becker, T., Anderson, J., Lysaght, P.: Modular dynamic reconfiguration in Virtex FPGAs. IET Proc. Comput. Digital Tech. **153**(3), 157–164 (2006)
37. Cheng, L., Wong, M.D.F.: Floorplan design for multi-million gate FPGAs. In: Proceedings IEEE International Conference on Computer-Aided Design, pp. 292–299 (2004)
38. Garcia, D., Amory, A., Moraes, F., Lubaszewski, M.: A CAD tool for the integration on hardware IP blocks. In: Proceedings of South Symposium on Microelectronics, pp. 55–58 (2004)

Secure Licensing of IP Cores on SRAM-Based FPGAs

Li Zhang and Chip-Hong Chang

Abstract The rapid increase in FPGA devices' capacity has enabled the implementation of complete sophisticated systems. The widening design productivity gap and shrinking time-to-market window have made licensing of external IP cores for system development pervade. The upfront IP licensing model currently used in the market is not a good fit to the FPGA IP market as the blanket IP license fee is too expensive for the majority of FPGA-based system developers who target low-to-medium volume applications. This chapter presents a detailed discussion on secure licensing of the IP cores that is more suitable for FPGA IP market. After the background information such as the stakeholders in the market, common attacks to the IP cores and desiderata for an effective licensing scheme have been presented, two categories of IP licensing schemes, i.e., those that employ only the conventional crypto primitives and those that exploit the emerging PUF primitive, are detailed and illustrated. The shared target of these schemes is to realize a secure pay-per-use licensing model where the system developer is proportionally charged for the use of the IP core. Along with the discussion of the schemes, some noteworthy features are also highlighted such as the authenticity of the IP core as well as the fault tolerance, tamper and side-channel resistance of the employed hardware primitives. The aim of this chapter is to provide a comprehensive overview of existing developments in FPGA IP licensing protocols and facilitate the development of new schemes that are more efficient and cost-effective.

L. Zhang • C.-H. Chang (✉)
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore
e-mail: lzhang2@e.ntu.edu.sg; echchang@ntu.edu.sg

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_12

1 Motivation of Secure Licensing Schemes for Core-Level FPGA IP Protection

1.1 Upfront Versus Pay-Per-Use IP Licensing Models for FPGAs

First introduced in the 1980s, Field Programmable Gate Array (FPGA) devices offer a ground-breaking alternative to static Application Specific Integrated Circuit (ASIC) solutions. Comparing to the mask programmable ASIC, the FPGA has significant competitive advantages in the flexibility to change or update the implemented functionality and the much lower non-recurring engineering (NRE) cost. At the early stage, FPGA devices are merely used for simple glue logic implementation. With the steady improvements in process technology and device architecture, the capacity of FPGA has skyrocketed to make highly sophisticated complete system implementable. For example, the Vertex-7 2000T FPGA announced by Xilinx in 2011 contains 6.8 billion transistors and provides 2 million logic cells, which is equivalent to 20 million ASIC gates, for user configuration [1]. Such capacity is more than satisfying the requirement for most system integration, ASIC prototyping and even replacement. Indeed, FPGAs have been widely used in telecommunications, networking, consumer, automotive and industrial applications, etc [2].

The widening of design productivity gap as a result of the rapid increase in the capacity of FPGA devices follows the same trend as the ASIC development. Under the even more demanding time-to-market window, it is no longer viable to design advanced FPGA systems from scratch. To shorten the design turnaround time, reuse-based design methodology has been prevailing, wherein reusable design blocks, also known as intellectual property (IP) cores, are purchased from external entities and incorporated into the system to be designed. This design method also allows the system designer to focus on the critical blocks of the system to differentiate their system from the competitive counterparts. Such modular design approach has been well supported by mainstream FPGA design tools. Adoption of standardized IP interface such as AXI4 interconnect [3] has paved the way for easy usage of third party IP cores. It facilitates the transaction of IP cores which in turn proliferates the FPGA IP development. In this market, the IP vendors provide reusable IP cores that are well optimized, rigorously tested and verified, and simple to be customized and implemented, while the system developers purchase the needed IP modules for use in their system designs.

Despite the general consensus and interest on reusing IP cores for the FPGA system development, the trading of FPGA IP cores is still far from getting up to speed with the transactions taken place in the ASIC IP market [4]. Two major concerns, if left unattended, will continue to hinder the growth of FPGA IP market. The first concern is the security of IP cores. It usually requires a large investment of money, time and effort to develop an IP core. The return of this investment is recovered from granting the usage of the IP core to the system developers through IP licensing. However, this return can be easily extorted as the bitstreams of unguarded or inadequately protected IP cores are vulnerable to misappropriation, reverse engineering and piracy by malicious licensees and adversaries. The second concern is the compatibility of licensing model for FPGA based IP cores. Due to the difficulty for the core vendor to monitor how many times a system developer uses the IP core to configure the FPGA chips, most core vendors adopt the upfront licensing model, in which the developer pays a vendor a lump sum of money for an unlimited use of the IP core. An expensive blanket license of unrestricted IP usage makes sense for ASIC products as the costly license fee can be amortized over the mass production of ASIC chips. However, FPGAs are preferred to ASICs mainly for low to medium volume applications. Such exorbitant licensing expenses eat directly into the profits or reduce the product pricing competitiveness of most FPGA-based system developers. It turns out that the upfront licensing model is a poor match to the FPGA IP market.

Secure licensing of FPGA based IP cores suitable for the low to medium volume applications is imperative to sustain the development of FPGA market. The desired licensing model should enable a core vendor to control or even monitor the number of times an IP core is used by the licensee. The transaction environment must assure the IP vendors that their IP cores cannot be abused and no design information will be divulged. Meanwhile, the system developers must also have the assurance that the received IP cores are authentic and uncompromised.

We will discuss how to realize such a secure licensing scheme for FPGA based IP cores in this chapter. Our discussion will be limited to IP cores targeting the SRAM-based FPGA. This FPGA type has the widest range of applications due to its denser construction, lower cost and better performance than other types such as the anti-fuse based and flash-based devices. In 2012, the SRAM-based FPGA accounted for 76.1 % of the global sales revenue and its market dominance is expected to continue for years to come [5]. On the other hand, the configuration bitstream of the SRAM-based FPGA is also the most vulnerable target of attacks due to the fact that the SRAM memory is volatile and the bitstream need to be stored in an external non-volatile memory (NVM). In the remainder of this chapter, the term FPGA refers to the SRAM-based FPGA unless explicitly stated otherwise.

1.2 Detective and Preventive Core-Level Protection Methods

Legal means such as the patent protection act and copyright law are feeble in fostering the desired secure licensing environment. They must be fortified by additional measures. In fact, there exist a large number of proposals for protecting the FPGA based designs. For example, one can simply move the external NVM into the package of the FPGA chip or even integrate the NVM onto the die of the FPGA [6]. In [7, 8], an extra secure device is incorporated into each FPGA chip. The design bitstream contains the same secret key and keyed hash function as the secure device, and its functionality will only be enabled after a successful authentication with the secure device through a challenge-response mechanism. In a similar method, the design reads the identity of the FPGA device (e.g. device DNA [9]) and processes it with the keyed hash function and secret key to generate a code to be compared with a secret code stored in an external secure NVM. Another popular protection method is to add a hardwired decryption engine and a non-volatile key storage component onto the chip. This approach ensures that the bitstream stored in the external NVM is in the encrypted form and can only be decrypted and used by the FPGA chip with the correct key. It is later enhanced with the authentication feature [10] to cope with the possible malicious tampering. Besides the specific demerits for each of the abovementioned methods, such as ease of being compromised, extra manufacturing cost and/or requirement of additional hardware device, the methods face the same limitation that the FPGA system can only be protected as a monolithic IP. As the system developer has full access to the content of the system, the upfront licensing model is assumed for the embedded IP cores. In other words, these methods are incapable of allowing the core vendor to control the number of times their IP cores are used. Such a limitation is also shared by the protection schemes in [11–13].

Secure licensing of the FPGA IP cores requires the core-level protection for the IP cores to be provided in the first place. Current core-level protection methods for FPGA IP cores can be divided into detective and preventive types. FPGA IP watermarking [14–16] and fingerprinting [17–19] techniques fall in the category of detective methods. These techniques embed a specific tag into the IP core by making use of its intrinsic feature or architecture. The tag may be the same for all IP instances and carry only the ownership information (i.e., watermarking), or it may also contain different user information in the IP instances distributed to different users (i.e., fingerprinting). The deficiency of these techniques is that they just passively confirm the occurrence of IP infringement. In contrast, the preventive methods effectively impede the IP infringement by making it more difficult and costly to succeed. According to the characteristics of existing preventive methods, they can be further divided into two subcategories based on their reliance on either the conventional crypto primitives or the physical unclonable function (PUF). PUF is an emerging security primitive that exploits the native and unique variations in the physical properties of the devices induced by the manufacturing process. Several interesting and effective FPGA IP licensing protocols have been developed based on PUF. A brief discussion of the PUF characteristics will be provided in Sect. 3.

This chapter will also discuss in details the preventive proposals for establishing the desired IP licensing environment.

2 The FPGA IP Market

2.1 Principals in the Market

The IP *core vendor* (CV) and the *system developer* (SD) are the seller and buyer of the market respectively. Besides these two principals who are directly involved in each IP transaction, another entity who participates indirectly in this process is the *FPGA vendor* (FV). The FV is the party who designs and sells FPGA chips. The bulk of FPGA fabrics are left uncommitted for the use by the SD. Nonetheless, it is also in the interest of the FV to include in some of their product families hard-wired functional primitives desired by most SDs to boost the chip sales. In general, the device architecture is different for the FPGAs purchased from different FV's, and even for the FPGAs from different device families of the same FV. The CV usually targets his design solutions to specific device families of the chosen FV's for optimized circuit performance and sells them as IP cores. When a transaction between the CV and the SD is committed, the IP core is usually customized according to the implementation requirements of the SD before the SD uses it in his system design. Using externally purchased IP cores and those designed in-house, the SD will then build a complete application, implement it on FPGA chips purchased from the FV, and sell the chips as producer goods or end products.

With the continuous scaling of the process nodes used for producing the FPGA devices, e.g., Xilinx 20 nm UltraScale devices [20], building and maintaining the corresponding state-of-the-art fabrication facility requires a humongous running expense. In fact, these fabrication facilities have been described as the most costly factories ever built by mankind [21]. The capital investment cost drives most FPGA vendors, e.g., Xilinx and Altera, to adopt a fabless manufacturing model and outsource the chip fabrication to external contracted fabs, called the *hardware manufacturer* (HM) here. In this business model, the FV pays the HM an upfront cost for the creation of the mask and for the required number of chips to be produced. However, a new problem emerges: although it is expensive and complicated to build a mask for the FPGA chip, the cost of producing extra chips is marginal for the HM after the mask has been created; the HM may produce excess chips and sell them in the grey market at a much lower price than the chips sold by the FV. This will greatly undermine the market revenue and deteriorate the brand name of the FV.

The activities of the four parties mentioned above and the interactions among them are depicted in Fig. 1. Besides these four parties, there exists an additional party in the market, known as the *malicious attacker* (MA), who steals the valuable IP cores or the design information. An MA is not necessarily a stand-alone party. It can also be acted by some SDs. The possible attacks that may be carried out by the MA will be discussed in details in Sect. 2.2.

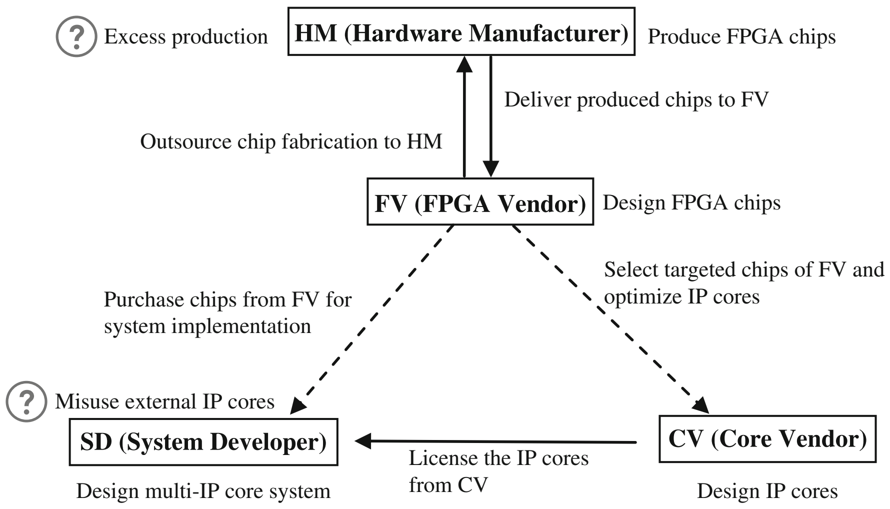

Excess production
HM (Hardware Manufacturer)
Produce FPGA chips
Deliver produced chips to FV
Outsource chip fabrication to HM
FV (FPGA Vendor)
Design FPGA chips
Select targeted chips of FV and optimize IP cores
Purchase chips from FV for system implementation
Misuse external IP cores
SD (System Developer)
CV (Core Vendor)
License the IP cores from CV
Design multi-IP core system
Design IP cores

Fig. 1 Interactions and interests of the four main parties in the FPGA IP market

2.2 *Common Attacks to FPGA Based IPs*

As mentioned earlier, with the SRAM memory being volatile, designs for the SRAM-based FPGA are vulnerable to attacks. The bitstream is easy to be poached by eavesdropping on the bus connecting the FPGA chip and the external NVM. If in plaintext, the bitstream can be readily duplicated and implemented on uncommitted FPGA chips bought off-the-shelf. Such an attack is called *cloning*. Cloning of unprotected bitstream is straightforward and incurs almost zero cost.

In addition to directly cloning the design, an attacker may also *reverse engineer* [22, 23] the poached bitstream to extract the proprietary design information, and work out a competitive design at a much lower cost. The effort required for reverse engineering a design is not as trivial as that for cloning. Nonetheless, tempted by the market value of the design information, the price of which is significant and inestimable sometimes, the attacker will have enough incentive to conceive such an attack.

The bitstream may also be *tampered* by the MA. For example, an attacker may tamper the bitstream in the set-top box, which controls the channels that the viewer can see, to bypass or even remove the security features to enjoy the services for free [24]. By tampering the bitstream, an attacker may also implant his own malicious logic, widely known as the hardware Trojan (HT) [25], into the design. The goal can be to access data stored in the FPGA, obtain more knowledge of the overall system, or even to hijack the system [10]. Due to the disastrous consequences of malicious tampering, rigorous integrity check is necessary for the third-party IP cores, especially when they are used in sectors like military, finance, energy and politics.

In addition, physical attacks are also commonly used by the MA. This kind of attack is capable of nabbing secret information by analyzing hardware characteristics of the FPGA that implements the target system. Physical attacks can be invasive or non-invasive. Invasive attacks are exemplified by the fault attacks [26, 27], which invade the device, tamper it for a desired fault and then analyze the difference between the correct and faulty outputs to extract sensitive information like the secret key. Powerful invasive attacks may involve sophisticated tools such as mechanical probes, focused ion beam (FIB) and scanning electron microscope (SEM) for studying the inner circuit structure of the chip. On the other hand, typical forms of non-invasive attacks are side-channel attacks, which infer the secret information from the side-channel information generated while the data is being processed. The side-channel information commonly exploited are computation time [28], consumed power [29] and emitted electromagnetic radiation [30].

2.3 Desiderata of FPGA IP Licensing Scheme

Acknowledging the risks of the activities and interactions of the principals in the FPGA IP market and the common attacks to the IP core, it is high time to consider what makes a good IP licensing scheme. The first two aspects that can be thought of are usually the security level that the licensing scheme can provide to IP cores and the cost to implement the scheme. Then, what is the desired security level for the IP cores? What is the reasonable cost? To answer these two questions, the analysis of security threats to electronic transaction systems performed by IBM may shed some valuable light. The security levels for modern electronic systems are classified as follows [31]:

- **Zero**: No special security features are added to the system.
- **Low**: Some security features are in place. They are relatively easy to be defeated with common laboratory or shop tools.
- **MODL**: More expensive tools are required, as well as some specialized knowledge. Tool cost may be ranging from \$500 to \$5000. The attack may become time-consuming but will eventually be successful.
- **MOD**: Special tools and equipment are required, as well as some special skills and knowledge. The tools and equipment may cost from \$5000 to \$50,000. The attack may be time-consuming but will eventually be successful.
- **MODH**: Equipment is available but is expensive to buy and operate. The cost involved may range from \$50,000 to \$200,000 or more. Special skills and knowledge are required to utilize the equipment for an attack. Multiple operations may be required and several adversaries with complementary skills may have to work on the attack sequence. The attack could be unsuccessful.
- **HIGH**: All known attacks have been unsuccessful. Some research by a team of specialists is necessary. Highly specialized equipment is necessary, some of which might have to be designed and built. Total cost of the attack could be 1 million dollars or more. The success of the attack is uncertain.

Instead of the security level HIGH, a transaction security system is suggested to be designed to the level of MODH in [31]. Indeed, a system protected at security level HIGH is usually too expensive to be receptive in the market. There should be a good balance between the high security level and the system cost. Similar considerations apply to the FPGA IP licensing scheme. A licensing scheme that makes the IP cores immune to any attacks is not a judicious target. A practical licensing scheme should be the one that makes a successful attack too expensive to be compensated by the benefits obtained from the stolen IP cores; at the same time, the implementation cost for the scheme should be market viable. Hence, we do not think it is necessary for a practical IP licensing scheme to be immune to the expensive physical attacks that use sophisticated tools to study the inner structure of the chip to learn the design information. These attacks exploit the weakness of the silicon technology; when they are possible, it is very difficult to secure the implemented design against them [12]. Nonetheless, it is usually too expensive to perform such an attack. Besides requiring advanced tools, it also needs special skills and tacit knowledge with a large amount of time and effort due to the size and complexity of the valuable IP cores and the FPGA chip today. Based on these considerations, we derive the following quality attributes that are essential for a good IP licensing scheme:

- **Broad Protection**: The IP cores should be well protected against all the perceivable low to medium cost attacks such as cloning, reverse engineering, malicious tampering and common physical and side channel attacks.
- **Low Implementation Cost**: The cost of enforcing the licensing scheme should be relatively low. The scheme should not complicate the already-complex manufacturing process. The use of new ancillary hardware, if any, should be well justified.
- **Transparency**: The scheme should not pose any change to current CAD tools. Neither does it impose any difficulty in reconfiguring or updating the designs in the chip, nor does it impact the chip reliability.
- **Cryptographically Secure**: The secret information should be well protected. The cryptographic algorithms used in the scheme, if any, must be widely accepted as secure.

3 Common FPGA Features and Notations

Before discussing the existing IP licensing schemes, it is helpful to give a brief explanation of the commonly employed device features. As the notations used for the same operation in different schemes may differ, the notations we are going to use in this chapter will also be provided along with the exposition.

As discussed in Sect. 2.3, low implementation cost is one of the most important desiderata for an FPGA IP licensing scheme. An intuitive approach is to exploit existing features in FPGA devices as much as possible and to reduce the demand for additional hardware primitives. The existing features that are commonly used by the licensing schemes are listed and explained as follows:

Device identifier The device identifier uniquely identifies each FPGA device. It can be a printed serial number or a bit string that is stored on chip. For example, on some Xilinx chips, a 57-bit device DNA is hardwired and can be read from the DNA port [9]. The device identifier for an FPGA device i is denoted as $\#ID_i$.

On-chip NVM and decryption engine In SRAM-based FPGA devices that support bitstream decryption, the secure NVM used for secret key storage is usually designed to be only accessible by the hardwired symmetric key decryption engine due to security concerns. The decryption engine is also not allowed to use an alternative key and can only be accessed by the configuration controller. A commonly used standard for the decryption engine is Advanced Encryption Standard (AES) [32]. An IP core in plaintext is simply denoted as IP_j and its identity as $\#IP_j$. When it is encrypted with a secret key K_{ENC}, the encrypted IP core is denoted as $E\ (K_{ENC}: IP_j)$.

Keyed-hash message authentication code (HMAC) As mentioned in Sect. 2.2, the bitstream may also be maliciously tampered, which may cause disastrous consequences. Although cyclic redundancy check (CRC) codes have been used in all bitstream formats to prevent faulty bitstream from being configured, they are not adequate to detect intentional malicious changes [33]. Some contemporary FPGA types have offered additional integrity check mechanisms that are more secure than the CRC code. For example, on Xilinx Vertex 6 and 7 devices the bitstream encryption/decryption feature is used in tandem with the SHA-256 HMAC [34] based authentication feature [35]. The encrypted and authenticated bitstream is generated by encrypting the plaintext bitstream, an HMAC key and the HMAC digest of the bitstream with the secret device key. The decrypted bitstream will only be configured on chip after its successful verification with the HMAC digest. The HMAC digest of IP_j under the key K_{MAC} will be denoted by $HMAC\ (K_{MAC}: IP_j)$. If no key is used, the generated hash digest is simply denoted as $h(IP_j)$.

Self-reconfiguration In recent FPGA devices, the configuration controller can be accessed from the fabric through an internal configuration access port (ICAP). This enables the FPGA logic to control the reconfiguration of part of itself. As will be seen in the discussion of some licensing schemes, the feature of self-reconfiguration may be used to remove the burden of using auxiliary modules to direct the IP decryption and installation. Upon completion of their missions, these modules will be erased from the fabric to release the reconfigurable logic it occupied.

In recent years, a number of IP licensing schemes based on the **PUF primitive** have also been devised. As mentioned in Sect. 1, PUF is an emerging device-level primitive. Essentially, all PUFs can be considered as a mapping function between the input stimulus (i.e., challenge) and the output response. As the variations due to random process-related factors are unique for each PUF (e.g., the transistor gate thickness and the doping concentration of the channel region for the CMOS based PUF), the challenge-response pairs (CRPs) of the PUF on each device vary significantly and such behaviors cannot be duplicated in another device nor physically re-fabricated. A secondary side effect is the behavior of the PUF will change substantially upon modification, which means that the attackers cannot tamper the PUF to gain access to the protected information. This property ensures that the response R of the CRP is inherently secured within the hardware device by means of its construction as opposed to its secure storage. For detailed discussion on the PUF, we refer the interested readers to [36] and the related chapters in this book. Common PUF primitives used on FPGA devices can be divided into the weak PUF and strong PUF types. The weak PUF is characterized by its possession of very few CRPs (or even just one CRP). Its typical application is to establish a unique device secret key. Being physically unclonable and tamper-resilient, the weak PUF replaces the less secure digital secret key stored in on-chip NVM to provide unique identification of the device. One good example of the weak PUF is the SRAM-type PUF on SmartFusion2 SoC FPGA [37]. At the power up of an SRAM cell with no write operation, the small transistor threshold mismatches will trigger the cell's positive feedback loop and make the cell assume one of the two possible states (0 or 1). For each cell, this start-up state value can be repeated with a high probability. However, the start-up behaviors of different cells are random and uncorrelated at large. By employing the random start-up behavior of a 16-Kbit SRAM block, the PUF primitive can establish unique secret keys for each FPGA device.

In contrast to the weak PUF, a strong PUF is characterized by its possession of a large number of CRPs. Exhausting all its possible challenge-response pairs within a limited time span (days or weeks) is impossible. From this perspective, the native response to a specific challenge needs not be encapsulated internally, by hash function for example, to prevent its exposure to the outside world. Strong PUF can be used to establish advanced protocols such as key exchange, zero-knowledge proof, oblivious transfer and bit commitment. This characteristic enables the creation of some very interesting and effective licensing schemes.

As analyzed in [38], the PUF response R cannot be straightforwardly used as a cryptographic key due to the noise in R and its non-uniform distribution. A fuzzy extractor is usually associated with the PUF primitive to generate the helper data W to correct the error, extract the randomness and finally establish a stable key K. To simplify our presentation of the licensing schemes using strong PUF, we assume that the helper data W is always provided with the corresponding challenge C and that the response R actually represents the established key K. That is to say, the response R, which is essentially the key K, is ready for use in message encryption and decryption.

4 Conventional Crypto Primitives Based Licensing Schemes

4.1 Pay-Per-Use Based FPGA IP Licensing Scheme

The first proposal to resolve the mismatch between the upfront FPGA IP licensing model and the IP core budget for low-to-medium-volume applications appeared in 2002. A creative model [39], wherein the FPGA IP cores are licensed on a pay-per-use basis, was devised and enabled in the method. The two prominent benefits of this new business model are: (1) the system developer with an application of low to medium volume no longer needs to pay the same expensive license fee as a developer with a high-volume application. (2) Charging for the use of the IP core is proportional to the sale of the system developed, reducing the stake of the system developer in the relentless market competition. With these advantages, the pay-per-use model becomes the common target of almost all the latter methods [38, 40–44]. The details of the scheme are described below.

Besides the FV, CV and SD described in Sect. 2.1, a *trusted third party* (TTP) is also involved in the digital rights management protocol. The TTP is an organization that all parties in the FPGA IP market trust for managing the secret information related to the IP transactions. In this scheme, the FV is assumed to design and manufacture the FPGA chips, i.e., taking the roles of both the FV and the HM. The protocol requires that each FPGA contains a permanent secret device key, a symmetric cipher for both bitstream encryption and decryption, and a unique device identity that is available on request through the programming interface. The protocol consists of three phases, which is depicted in Fig. 2.

In the first phase, which is the *enrollment of FPGA devices*, *security tokens* are created by encrypting *user keys* with each FPGA's secret device key. Any entity that has the possession of the chip can create the user key and security token pairs. Although the author suggests that the security tokens of all FPGAs can be created at the time of manufacture, e.g., during final testing of the FPGAs, to minimize the cost, better security will be achieved if the security tokens are created at the site of the TTP. The TTP will then store the pairs of user key and token in his database indexed by the chip identifier, which will later be used for billing of the IP usage and creation of device specific encrypted bitstreams.

In the phase of *IP usage by the SD*, the SD obtains the needed IP cores from the CVs. The IP cores are in encrypted form so as to prevent misuse or tampering by the SD; but they can be decrypted by the FV's CAD tool to allow processing. This can be realized with the public key cryptographic algorithm like RSA and ECC: the IP cores are encrypted with the public key PK_{CAD}. The encrypted IP cores can be decrypted by the private key SK_{CAD} embedded in the CAD tool. After the SD has completed the system design, the CAD tool will issue a *design key* to encrypt the bitstream of the system. The bitstream is different from the conventional one in that it contains the copyright information of all the licensed IP cores. It can be decrypted by the programming tool in the next phase, which has access to the secret

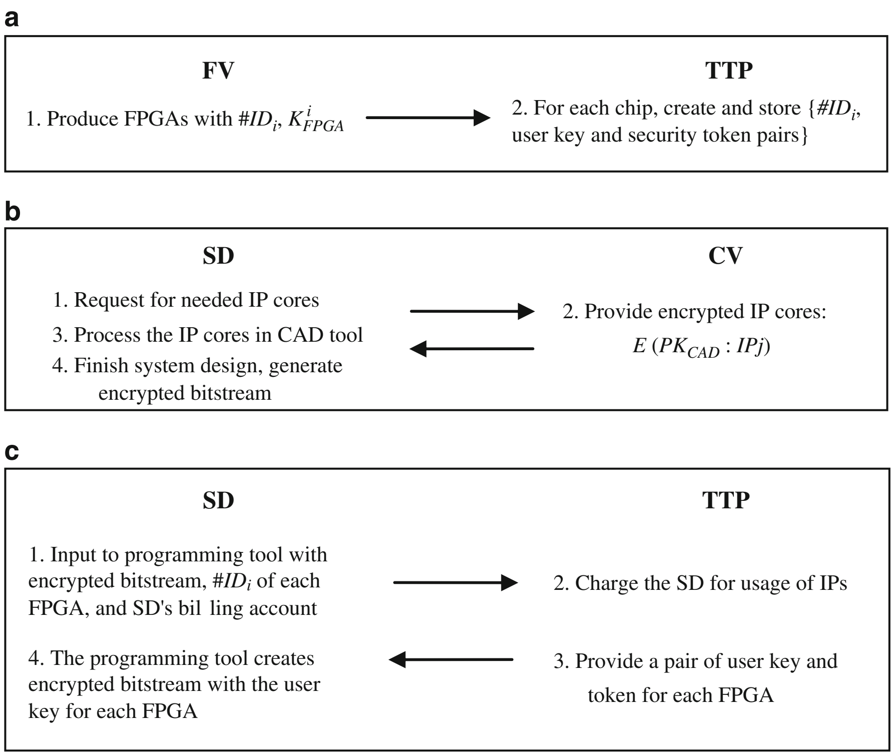

a
FV
TTP
1. Produce FPGAs with $\#ID_i$, K^i_{FPGA}
2. For each chip, create and store {$\#ID_i$, user key and security token pairs}
b
SD
CV
1. Request for needed IP cores
3. Process the IP cores in CAD tool
4. Finish system design, generate encrypted bitstream
2. Provide encrypted IP cores: $E\ (PK_{CAD} : IPj)$
c
SD
TTP
1. Input to programming tool with encrypted bitstream, $\#ID_i$ of each FPGA, and SD's bil ling account
2. Charge the SD for usage of IPs
4. The programming tool creates encrypted bitstream with the user key for each FPGA
3. Provide a pair of user key and token for each FPGA

Fig. 2 The first pay-per-use based FPGA IP licensing scheme proposed in [39]. (**a**) Enrollment of FPGA devices. (**b**) IP usage by the SD. (**c**) Creation of device specific encrypted bitstreams

information stored in the server of the TTP. Again, the bitstream encryption in the CAD tool and its decryption in the programming tool can be realized with the public key cryptography.

In the last phase, device specific encrypted bitstreams for the FPGAs are created based on the encrypted bitstream generated from the CAD tool. The programming tool takes as input the encrypted bitstream from the CAD tool, the IDs of the FPGAs to be programmed and the billing information of the customer, decrypts the bitstream with the stored secret information, and then charges the customer and creates the device specific encrypted bitstream. To perform the billing and encryption process, the programming tool is connected with the server provided by the TTP over the internet. Then the TTP's computer bills the customers and looks up in its database the chip ID to find the corresponding user key and token pairs. Next, the TTP sends one pair of the user key and security token to the programming tool, which will then encrypt the bitstream with the secret user key. Appended with the security token, the encrypted bitstream that can only be configured on a particular FPGA is created. As security critical information is transmitted between the programming tool and the TTP's server, the communication between them must be secure and is suggested by the authors to be protected by standard internet security protocols.

The chip configuration using the device specific encrypted bitstream is kept relatively simple. When the FPGA loads the bitstream, it first recovers the user key from the security token with its secret device key. With the user key, the bitstream is then decrypted before it is used for chip configuration. There are two main concerns for the proposed scheme. One is the requirement of hardwired circuitry for both the encryption and decryption, but only the on-chip decryption engine is available on the commercial devices. The other is that both the CAD tool and the programming tool contain secret information. These secret information need to be carefully protected. Otherwise, a successful crack to any of these tools will cause the leakage of the IP content. Recognizing this risk, the author suggests some variants of the scheme in [45], where the tools run on the TTP's server. With no secret information contained in the tools at the site of the SD, these alternatives provide better security. But the price to be paid is that the TTP needs to be equipped with much more computational power and higher internet connection bandwidth.

In most cases, the higher level the protected IP cores reside, the more parties and design tools are involved, and the more complicated is the task of protecting the IP cores. As a result, the authors of [40] suggest that the most suitable target for core-level FPGA IP protection is the bitstream, which is generated at the last stage of the FPGA design and implementation flow. Indeed, if the IP cores are communicated at the level of bitstream, it is possible to set the security boundary at the FPGA chip, i.e., the IP cores can be kept in encrypted form after it leaves the site of the CV and before it is configured on the chip.

4.2 Public-Key Crypto Based Key Derivation Function

The pay-per-use FPGA IP licensing scheme proposed in [40] is actually an extension of the scheme in [13], which is only capable of providing solution to licensing a single monolithic FPGA design. Compared with the setup of the scheme in [39], a key derivation function (KDF) module based on public-key cryptography is added, which helps to securely transport and install the needed key for bitstream decryption. One novelty of the scheme is that it avoids the burden of the public-key functionality, which is usually resource-consuming, by moving the KDF module to a temporary configuration bitstream. After establishing the bitstream decryption key, the resources occupied by the KDF module can be released for use by the IP cores or other system blocks. Out of the four stages of licensing protocol described in [40], the first two are for generating the device specific bitstreams. These two stages are depicted in Fig. 3.

In the security information setup stage, the FV generates a secret device key K^i_{FPGA} and an asymmetric key pair $(SK^i_{FPGA}, PK^i_{FPGA})$ for FPGA batch i. The size of the FPGA batch needs to be kept relatively small so as to limit the damage if

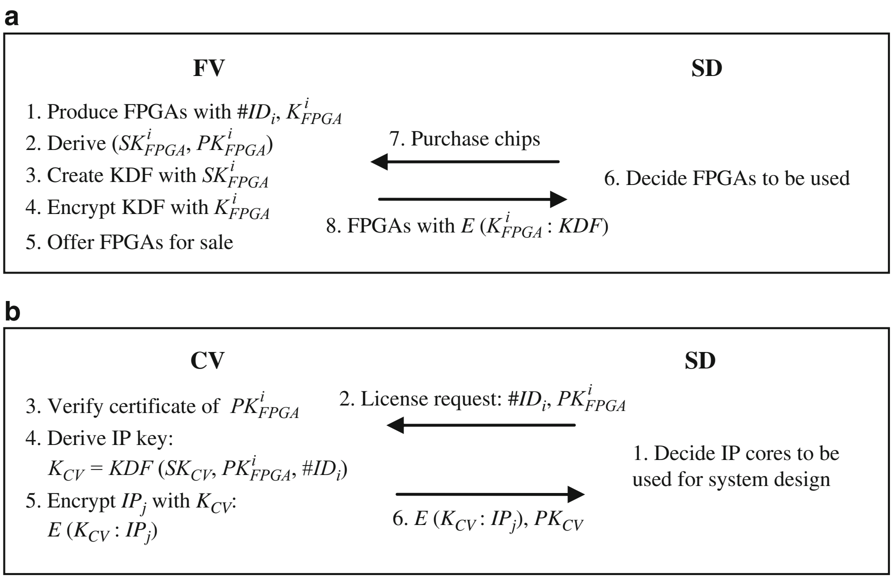

a
FV
SD
1. Produce FPGAs with $\#ID_i$, K^i_{FPGA}
2. Derive (SK^i_{FPGA}, PK^i_{FPGA})
3. Create KDF with SK^i_{FPGA}
4. Encrypt KDF with K^i_{FPGA}
5. Offer FPGAs for sale
7. Purchase chips
8. FPGAs with E (K^i_{FPGA} : KDF)
6. Decide FPGAs to be used
b
CV
SD
3. Verify certificate of PK^i_{FPGA}
4. Derive IP key:
$K_{CV} = KDF\ (SK_{CV}, PK^i_{FPGA}, \#ID_i)$
5. Encrypt IP_j with K_{CV}:
$E\ (K_{CV} : IP_j)$
2. License request: $\#ID_i$, PK^i_{FPGA}
6. $E\ (K_{CV} : IP_j)$, PK_{CV}
1. Decide IP cores to be used for system design

Fig. 3 FPGA IP licensing using public-key crypto based key derivation function [40]. (**a**) Security information setup stage. (**b**) IP licensing stage

the secret key K^i_{FPGA} or SK^i_{FPGA} is compromised. For brevity, we assume one chip per batch in our description. At the same time, the FV creates a specific bitstream of the KDF which will be used to derive the IP core decryption keys. The KDF bitstream, which contains the private key SK^i_{FPGA} of the key pair, is the vital part of the proposed scheme and is encrypted with the secret device key K^i_{FPGA}. Hence, in the scheme, each FPGA chip bought by the SD will be associated with one encrypted KDF bitstream.

When the SD wants to license the usage of one IP core from the CV on his purchased FPGA chips, he will send his licensing request to the CV together with the device ID $\#ID_i$ and the public key PK^i_{FPGA} of each chip i. After receiving the request, the CV will verify the certificate of PK^i_{FPGA} to confirm that it is originated from the FV. Upon a successful verification, the CV will create the key for IP encryption and decryption K_{CV}: He first generates an asymmetric key pair (SK_{CV}, PK_{CV}) for the specific SD and then derives K_{CV} as in Eq. (1).

$$K_{CV} = KDF\left(SK_{CV}, PK^i_{FPGA}, \#ID_i\right) \tag{1}$$

The IP core may be first tailored to the SD's implementation requirements before it is encrypted with K_{CV}. After charging the SD for the license fee, the specifically encrypted bitstream for FPGA chip i together with the CV's public key PK_{CV} is transferred to the SD.

At his site, the SD will stitch the external IP blocks and his own functional blocks based on the pre-defined bitstream-level partitioning. The authors propose that the bitstream of the IP core contain an additional command to locate the decryption key that has been derived and stored in specific key storage. Derivation of the IP decryption key K_{CV} is performed by the KDF bitstream, which is configured on chip after being decrypted by the device key K^i_{FPGA}. The key derivation is performed with SK^i_{FPGA} stored in the KDF bitstream, device ID *#ID* and the CV's public key PK_{CV}, as shown in Eq. (2).

$$K_{CV} = KDF\left(PK_{CV}, SK^i_{FPGA}, \#ID_i\right) \tag{2}$$

The secret device key K^i_{FPGA} and the asymmetric key pair (SK^i_{FPGA}, PK^i_{FPGA}), which are both decided by the FV, are the vital information for the security of protected IP core. Hence, the FV is assumed to be a trustworthy and unbiased TTP. The authors propose two versions of the specific key storage for the IP decryption keys K_{CV}'s. The first version uses secure non-volatile key storage memory. This way the key establishment process occurs just once and the KDF bitstream will no longer be needed for the future configuration of the system. The drawback of this method is that it requires additional such non-volatile key storage, which is not available in current commercial devices. The second version uses volatile key storage memory and K_{CV}'s are established every time after powering up the device. This version eliminates the need for the additional non-volatile key storage and generates K_{CV}'s only when needed. Nonetheless, the whole system configuration time may be longer and the KDF bitstream as well as the key establishing information (e.g., PK_{CV}) need to be stored in the external configuration memory. In addition, decrypting the IP core using K_{CV} by hardwired decryption engine is not allowed in currently available commercial FPGAs. The solution to this problem is by either adding a symmetric decryption engine in the KDF bitstream or modifying the on-chip decryption engine.

4.3 Trusted Third Party with Symmetric-Key Crypto Based Key Derivation Function

Later, a scheme [41] that is applicable to commercially available FPGA devices is created without incurring any hardware modification. Instead of assuming that the FV is fully trusted by other parties in the market, the scheme separates the role of the FV and the TTP. With an external TTP appointed in the licensing scheme, the asymmetric cryptography based KDF, which is used for secure key transportation and installation, becomes dispensable and is replaced by the symmetric cryptography.

After the blank FPGA chips are produced by the FV, they are sent to the site of the TTP for embedding the secret device key K^i_{FPGA}. The TTP also associates each FPGA chip with a core installation module (CIM) before sending the chips back to the FV. The CIM bitstream serves a similar purpose as the KDF bitstream

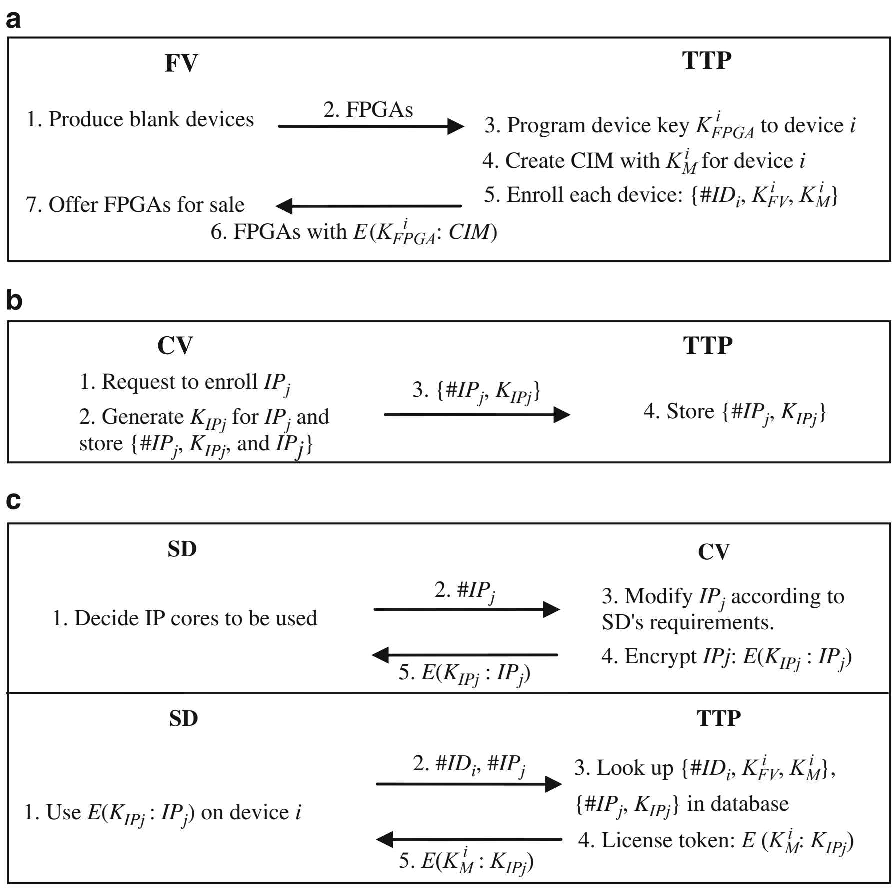

a
FV
TTP
1. Produce blank devices
2. FPGAs
3. Program device key K^i_{FPGA} to device i
4. Create CIM with K^i_M for device i
5. Enroll each device: $\{\#ID_i, K^i_{FV}, K^i_M\}$
7. Offer FPGAs for sale
6. FPGAs with $E(K^i_{FPGA}: CIM)$
b
CV
TTP
1. Request to enroll IP_j
2. Generate K_{IPj} for IP_j and store $\{\#IP_j, K_{IPj}$, and $IP_j\}$
3. $\{\#IP_j, K_{IPj}\}$
4. Store $\{\#IP_j, K_{IPj}\}$
c
SD
CV
1. Decide IP cores to be used
2. $\#IP_j$
3. Modify IP_j according to SD's requirements.
4. Encrypt IPj: $E(K_{IPj} : IP_j)$
5. $E(K_{IPj} : IP_j)$
SD
TTP
1. Use $E(K_{IPj} : IP_j)$ on device i
2. $\#ID_i, \#IP_j$
3. Look up $\{\#ID_i, K^i_{FV}, K^i_M\}$, $\{\#IP_j, K_{IPj}\}$ in database
4. License token: $E(K^i_M: K_{IPj})$
5. $E(K^i_M : K_{IPj})$

Fig. 4 The IP licensing scheme using symmetric crypto and TTP for recent FPGAs [41]. (**a**) Enrollment of FPGA devices. (**b**) Enrollment of IP cores. (**c**) IP core licensing

described above, i.e., to establish the IP decryption key on FPGA. The difference between them lies in: instead of the public-key crypto, the CIM of chip i contains one symmetric cipher and one dedicated secret metering key K^i_M. The metering key will be used for recovering the IP decryption key. The licensing protocol is depicted in Fig. 4.

The IP decryption process in this scheme uses the partial reconfiguration feature of the FPGA. The CIM bitstream is first decrypted with the secret device key K^i_{FPGA} by the on-chip decryption engine and configured in the user logic. The design in the CIM contains a symmetric cipher for decryption, an internal configuration access port (ICAP) interface and two key registers, as shown in Fig. 5. One key register holds the metering key K^i_M; the other one is empty initially and will be used to store the IP decryption key. The ICAP port will be used to access the configuration controller for configuring the decrypted IP cores. When the encrypted

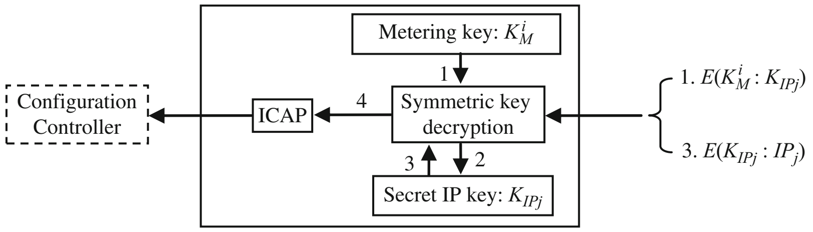


Fig. 5 The symmetric cryptography based core installation module

IP key $E(K_M^i : K_{IP1})$ for the first IP core is loaded, it will be decrypted by the metering key and saved in the empty key register of the CIM. Next, the first encrypted IP core E (K_{IP1}: IP_1) will be loaded onto the CIM, decrypted with K_{IP1}, and transferred to the configuration controller through the ICAP port for configuration on pre-defined reconfigurable logic area. The above configuration process will continue for all the remaining licensed IP cores, where the license of an IP core is issued as an IP decryption key K_{IPj} encrypted by the specific device metering key K_M^i of FPGA i.

4.4 Partially Trusted Third Party with Establishment Module

Concerning the expensive logistic required for the FV to transfer all his FPGA devices to and fro the TTP's secure site and the lack of an independent entity that is fully trusted by all the parties in the IP market, the scheme in [42] let the FV take the role of the TTP again. Nonetheless, the setup of the scheme does not rely fully on the FV to act honestly in discharging this duty. Instead, the scheme prevents the FV from having an easy access to the secret information related to the IP core. This is accomplished by transferring the ownership of the CIM provision from the FV to the CV of the respective IP cores. A similar setup is used in the protection scheme for multi-FPGA systems proposed in [46], which implements one IP core per chip though. In this way, the CV is capable of deciding the secret key K_{CVj} and the cryptographic components in the CIM all by himself. Besides, the CV can also refresh the CIM of his volition with an updated version using a different K_{CVj} and state-of-the-art cryptographic components. The secret key K_{CVj} is used to generate a secret device specific key K_{CVj}^i, which is used to protect the IP encryption key K_{IPj} and create a device specific license token T_{CVj}^i. K_{CVj}^i and T_{CVj}^i is generated by Eqs. (3) and (4), respectively. The SD's fingerprint, denoted by $F(\#SD)$, can also be inserted when the CV tailors the IP core to the SD's implementation requirement. The inserted fingerprint helps to further deter the SD from misusing the licensed IP cores. The licensing steps are depicted in Fig. 6.

$$K_{CVj}^i = h\left(K_{CVj}, \#ID_i\right) \tag{3}$$

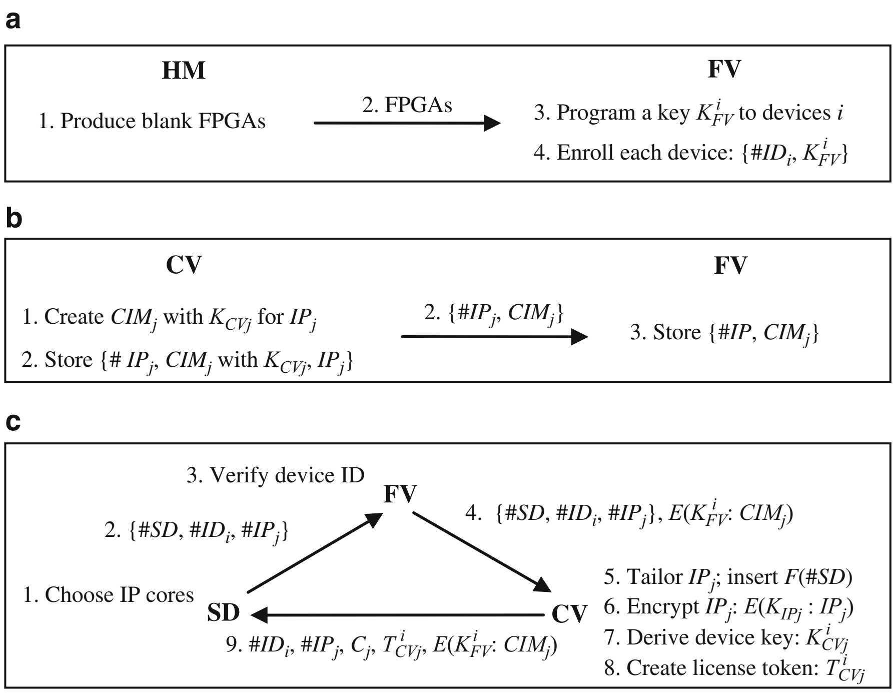

a
HM
FV
1. Produce blank FPGAs
2. FPGAs
3. Program a key K_{FV}^i to devices i
4. Enroll each device: $\{\#ID_i, K_{FV}^i\}$
b
CV
FV
1. Create CIM_j with K_{CVj} for IP_j
2. Store $\{\# IP_j, CIM_j$ with $K_{CVj}, IP_j\}$
2. $\{\#IP_j, CIM_j\}$
3. Store $\{\#IP, CIM_j\}$
c
3. Verify device ID
FV
2. $\{\#SD, \#ID_i, \#IP_j\}$
4. $\{\#SD, \#ID_i, \#IP_j\}, E(K_{FV}^i: CIM_j)$
1. Choose IP cores
SD
CV
5. Tailor IP_j; insert $F(\#SD)$
6. Encrypt IP_j: $E(K_{IPj} : IP_j)$
7. Derive device key: K_{CVj}^i
8. Create license token: T_{CVj}^i
9. $\#ID_i, \#IP_j, C_j, T_{CVj}^i, E(K_{FV}^i: CIM_j)$

Fig. 6 The pragmatic per-device IP licensing scheme proposed in [42]. (**a**) Enrollment of FPGA devices. (**b**) Enrollment of IP cores. (**c**) IP core licensing

$$T_{CVj}^i = E\left(K_{CVj}^i : K_{IPj}\right) \tag{4}$$

During the IP enrollment stage, the plaintext CIM is transferred from the CV to the FV. Although the communication channel between the CV and FV is assumed to be secure and authentic, the secure communication of the CIM may still be a concern. To circumvent this problem, the delivery of plaintext CIM is avoided in a strengthen version of this scheme. This is achieved by associating with each FPGA an establishment module (EM). The EM, which is encrypted by the device secret key K_{FV}^i, is provided by the FV. It consists of a public-key crypto based key derivation core, a symmetric decryption core and a unique private key SK_{EM}^i for device i. The public–private key pair (PK_{EM}^i, SK_{EM}^i) for each FPGA i is stored in the database of the FV and the public key PK_{EM}^i is delivered with the IP licensing request from the FV to the CV in message 4 of Fig. 6c. From the EM module, the CV can derive a shared secret key in the similar manner as in Eqs. (1) and (2). The shared key, denoted by K_{CVj}^{CIMi}, is used to encrypt the CIM module before the module is sent to the FV. The advantage of this enhancement is that all the important data related to the IP core, including the CIM module, are now in encryption form upon leaving the site of the CV and before they are configured on chip. As the same symmetric decryption core may be shared by the EM module and the CIM module, the additional cost of this enhanced version is mainly the public-crypto based key derivation core.

We would like to emphasize that *authentication* is equally important as *encryption* in an IP licensing scheme. Without verifying the authenticity of the received IP core, the FPGA may be configured with a design of any form and from anyone with the consequence of, at best, the denial of service, or in the worst case, the execution of unauthorized code [47]. In our description of the above schemes, only encryption of the secret information is mentioned for the ease of understanding. In actual fact, each decryption process is to be preceded by an integrity check step. For example, the SHA-256 HMAC based authentication is activated in tandem with the bitstream decryption on Xilinx Vertex 6 and 7 devices, as described in Sect. 3. In [42], the possibility of employing Keccak [48], the winner of the NIST secure hash algorithm (SHA-3) competition in 2012, is also conceived for bitstream authentication. Without the security hole of SHA-256, Keccak does not require the current nested approach to generate the HMAC digest, which will simplify the HMAC generation process and hence the cost of implementing the authentication feature. Besides, Keccak also provides the mode of authenticated encryption (based on duplex construction). Once standardized, Keccak can be a good candidate for realizing a compact authenticated encryption core.

Before moving to the next section, we would also like to highlight that all cryptographic components used in the licensing schemes need also be fault-tolerant and resilient against side-channel attacks. The various physical attacks have been described in Sect. 2.2. Interested readers are referred to [42, 49, 50] for a good evaluation of the cost for adopting various countermeasures against these physical attacks for the AES and HMAC core.

5 Physical Unclonable Function Based Licensing Schemes

The usage of the weak PUF is transparent to the non-PUF based licensing protocols described above, where the secret key stored in the on-chip secure NVM can be directly replaced by the weak NVM. The focus of this section is to discuss the licensing schemes which employ the strong PUF. In particular, these licensing schemes take advantage of the large number of CRPs possessed by the strong PUF and are very different from those discussed in the above section.

The first strong PUF based licensing scheme that is applicable to FPGA IP cores is proposed in [43]. In this scheme, each FPGA is assumed to contain a security module which consists of a PUF block and a decryption and authentication block (denoted as D&A block). After building the FPGAs, the FV enrolls a specific set of CRP's for each FPGA device. The CRP's, together with the chip identity $\#ID_i$ for each FPGA, is sent to an external TTP. At the same time, the CV also needs to enroll his IP cores into the TTP's database by sending the IP identity $\#IP_j$ and the IP authentication information, i.e., *Hash* ($\#IP_j$, IP_j), to the TTP. Both the

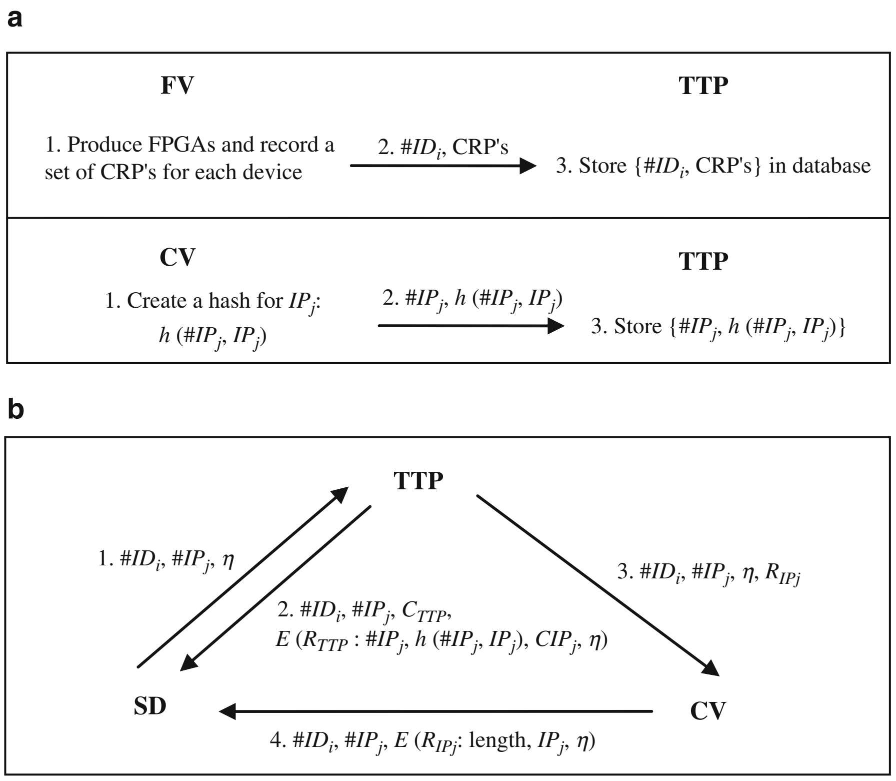

a
FV
TTP
1. Produce FPGAs and record a set of CRP's for each device
2. #ID_i, CRP's
3. Store {#ID_i, CRP's} in database
CV
TTP
1. Create a hash for IP_j: h (#IP_j, IP_j)
2. #IP_j, h (#IP_j, IP_j)
3. Store {#IP_j, h (#IP_j, IP_j)}
b
TTP
1. #ID_i, #IP_j, η
2. #ID_i, #IP_j, C_TTP, E (R_TTP : #IP_j, h (#IP_j, IP_j), CIP_j, η)
3. #ID_i, #IP_j, η, R_IPj
SD
CV
4. #ID_i, #IP_j, E (R_IPj: length, IP_j, η)

Fig. 7 The licensing scheme proposed in offline HW/SW authentication for reconfigurable platforms [43]. (**a**) FPGA device and IP core enrollments. (**b**) IP core licensing

communication channel between the FV and the TTP and the channel between the CV and the TTP are assumed to be secure and authenticated. The IP licensing steps are depicted in Fig. 7.

After deciding the external IP cores to be used in his system, the SD sends the identity $\#ID_i$ of the FPGA, which is to be used for system implementation, the identity $\#IP_j$ of the IP core and a nonce η to the TTP. For each IP core request, the TTP replies the SD with message 2, which contains the information needed to decrypt and authenticate the requested IP core. The IP decryption and authentication information, i.e., $\{\#IP_j, Hash\ (\#IP_j, IP_j), C_{IPj}, \eta\}$, is encrypted by the response R_{TTP} of a CRP, $\{C_{TTP}, R_{TTP}\}$, selected by the TTP. C_{IPj} is the challenge of another CRP $\{C_{IPj}, R_{IPj}\}$, which is to be used for the protection of the IP core. Message 3 is the relayed IP request from the TTP to the CV, which includes the response R_{IPj} of the CRP, $\{C_{IP}, R_{IPj}\}$, for encrypting the IP core. After receiving Message 4 from the CV, the SD will have the following two messages for the FPGA with identity $\#ID_j$:

$$C_{TTP}, \#IP_j, E\left(R_{TTP}: \quad \#IP_j, \quad h\left(\#IP_j, IP_j\right), C_{IPj}, \eta\right) \tag{5}$$

$$\#IP_j, E\left(R_{IPj} : length, IP_j, \eta\right) \tag{6}$$

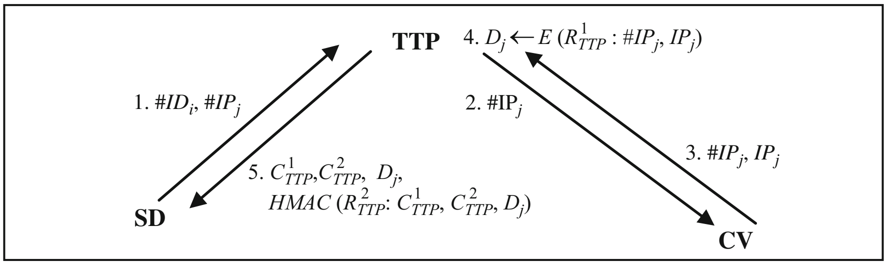
Only the PUF primitive on the specific FPGA device is able to generate the correct R_{TTP} under the challenge C_{TTP}. After the successful recovery of C_{IPj} and the IP authentication information using the correct R_{TTP}, C_{IPj} is used to generate the correct R_{IPj} which is then used to recover the IP core. Thus, the IP core will only be configured on chip after its successful authentication.

In the above scheme, the CRP of the PUF is used for the authentication of FPGA device and for protecting the secret information related to the IP core. As a strong PUF possesses a large amount of CRPs, different CRPs can be used for protecting different IP cores. This facilitates the simplification of communication between the TTP and the CV: the unique response of a specific FPGA's PUF upon a challenge can be directly provided to the CV for IP encryption. As long as the same CRP is not used for protecting the IP core of another CV, the security of the IP core is ensured. Due to this reason, communications involved in strong PUF-based licensing schemes are usually simpler than those schemes that employ conventional on-chip secret keys.

As the above protocol also aims for protecting the software IPs that run on the microprocessors configured on FPGA, the length information of the IP core is also provided in Message 4 of Fig. 7b. However, when dealing with the hardware IP designs in bitstream, separate length information is not necessary, as it is already embedded in the bitstream files for current FPGAs. Besides, the above protocol requires two decryptions and one hash computation to be performed in the on-chip D&A module before the IP configuration. The possibility to further simplify the protocol is pointed out in [38]. As the channel between the SD and the CV is unsecure, the TTP has access to the IP content with his knowledge of the CRPs. In this case, if the TTP is assumed to be fully trusted in the licensing scheme, the IP core can be directly provided by the CV to the TTP who will then encrypt the IP core IP_j with the response R^1_{TTP} of a selected CRP, $\{C^1_{TTP}, R^1_{TTP}\}$, and create the keyed authentication information with the response R^2_{TTP} of another CRP, $\{C^2_{TTP}, R^2_{TTP}\}$. This modified scheme (as depicted in Fig. 8a) will reduce the required operations that are performed on chip to one decryption and one MAC (hash). As this modified scheme requires the participation of TTP in each communication of the IP licensing process and each SD has to obtain the IP core from the TTP, it may create a system bottleneck. A variation of the scheme is also proposed. This variant is depicted in Fig. 8b, where the TTP provides the CRPs, $\{C^1_{TTP}, R^1_{TTP}\}$ and $\{C^2_{TTP}, R^2_{TTP}\}$, to the CV so that the CV can create the protected IP core on his own. Subsequently, the CV sends the protected IP core and the authentication information to the SD. The required on-chip operations are still maintained at one decryption and one MAC before the IP core can be configured, but the number of needed communications is reduced from four to three.

As mentioned in Sect. 4, an entity that is completely trusted by all the parties in the FPGA IP market is not available. Although the FV has some incentives to take the role of the TTP to facilitate the IP licensing and transaction, it is hard for the FV, as a business entity, to be fully trusted by all the CVs, especially those with IP cores

a


TTP
SD
CV
1. $\#ID_i, \#IP_j$
5. $C^1_{TTP}, C^2_{TTP}, D_j, HMAC(R^2_{TTP}: C^1_{TTP}, C^2_{TTP}, D_j)$
2. $\#IP_j$
3. $\#IP_j, IP_j$
4. $D_j \leftarrow E(R^1_{TTP}: \#IP_j, IP_j)$

b

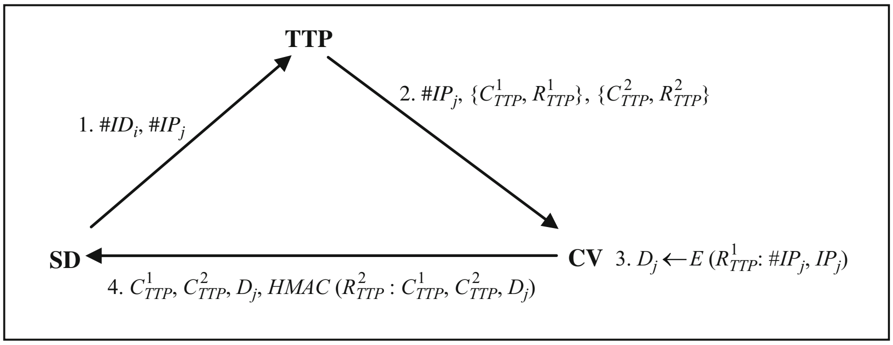

TTP
SD
CV
1. $\#ID_i, \#IP_j$
2. $\#IP_j, \{C^1_{TTP}, R^1_{TTP}\}, \{C^2_{TTP}, R^2_{TTP}\}$
3. $D_j \leftarrow E(R^1_{TTP}: \#IP_j, IP_j)$
4. $C^1_{TTP}, C^2_{TTP}, D_j, HMAC(R^2_{TTP}: C^1_{TTP}, C^2_{TTP}, D_j)$

Fig. 8 Two IP licensing stages of strong PUF-based protocol with fully trusted TTP [38]. (**a**) A simplified IP licensing stage with the TTP providing IP cores. (**b**) A simplified IP licensing stage with the CV providing IP cores

of significant values. Hence, a licensing scheme with the TTP that is assumed to be fully trusted and has access to the IP content may not be receptive. In view of this, the authors of [38] also proposes another variant, which is depicted in Fig. 9. Instead of assuming that the TTP is fully trusted by the participants, a so-called *honest-but-curious* model is used. The same assumption is also used in the licensing scheme [42] described in Sect. 4. To prevent the TTP from having access to the IP content, a random nonce η generated by the SD and a public–private key pair of the CV, (PK_{CV}, SK_{CV}) is used. In the request for IP license, the SD uses the public key PK_{CV} of the CV to encrypt the nonce that can only be recovered by the CV. Together with R^1_{TTP} and R^2_{TTP} provided by the FV, the nonce η will be used by the CV to devise the secret keys for the IP encryption and MAC generation.

Due to the public-key crypto used for transferring the secret information from the SD to the TTP and from the TTP to the CV, confidentiality of the secret information is guaranteed and only the communication channels between the SD and the TTP and between the TTP and the CV are required to be authenticated. Under the assumption of a honest-but-curious TTP, the TTP will not tamper with the encrypted nonce, i.e., $E(PK_{CV} : \eta)$. Comparing with the protocols in Fig. 8, where the on-chip operations needed before the IP configuration are merely one decryption and one MAC, two additional hash computations are required to generate the secret keys.

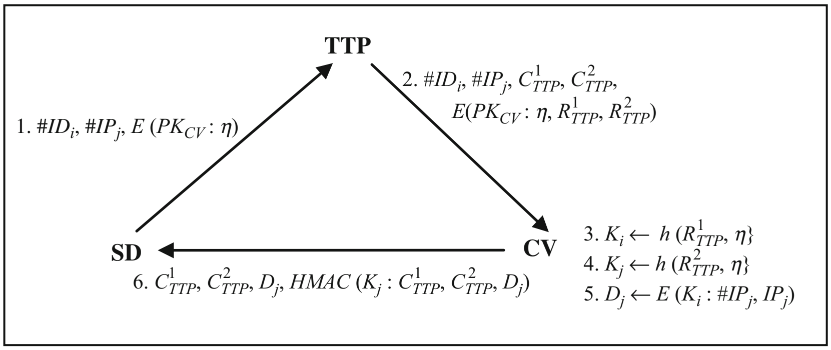

TTP
SD
CV
1. #ID_i, #IP_j, E (PK_CV : η)
2. #ID_i, #IP_j, C^1_TTP, C^2_TTP, E(PK_CV : η, R^1_TTP, R^2_TTP)
3. K_i ← h (R^1_TTP, η}
4. K_j ← h (R^2_TTP, η}
5. D_j ← E (K_i : #IP_j, IP_j)
6. C^1_TTP, C^2_TTP, D_j, HMAC (K_j : C^1_TTP, C^2_TTP, D_j)

Fig. 9 IP licensing with confidentiality of the IP content from the TTP [38]

Moving one step further, the same group of authors also proposed an even more secure licensing scheme which does not require the CRPs to be recorded in the enrollment stage. In other words, the response of the PUF is kept in the device and is not known to any party. The scheme is based on the public-key cryptography. Unlike the previous variant which employs the public-key crypto for nonce encryption and decryption, the scheme in [44] assumes an elliptic curve (EC) crypto based module to be hardwired on-chip. The EC module will generate asymmetric key pairs and perform IP decryption and authentication.

During the enrollment stage, the SD requires a certificate for the public key of the key pair generated by his FPGA device i. The key pair is generated as follows: the SD first obtains the domain parameters of the EC, such as the finite field F_{2^k} and EC point $P \in E(F_{2^k})$, published by the TTP. Then he injects a selected challenge C_1 to the PUF primitive of device i which will generate a response R_1. Based on R_1, the private key SK_i^1 is derived inside chip i; By computing $PK_i^1 = SK_i^1 \cdot P$, the public key PK_i^1 is generated and exported to the SD. Then the SD sends the information $\{\#ID_i, C_1, PK_i^1\}$ to the TTP and executes a zero-knowledge proof (ZKP) [51] that he is in possession of the specific device i which generates the private key SK_i^1 corresponding to the public key PK_i^1. After the successful verification, the TTP issues a certificate for PK_i^1. The certificate is generated by signing PK_i^1 using the TTP's private key SK_{TTP}, denoted by $Sig(SK_{TTP} : \#ID_i, C_1, PK_i^1)$. At the same time, the public key PK_{CV} of the CV undergoes a similar certification procedure to obtain $Sig(SK_{TTP} : \#IP_j, PK_{CV})$. With the above information established, the authentication and licensing of the IP core is analogous to a public-key crypto message transfer, as shown in Fig. 10.

The feasibility of this scheme relies on a convincing and robust zero-knowledge proof for certifying the public key of the FPGA devices and the CV. In particular, it must be assured that the public key PK_j^i provided by an SD is truly obtained from a certified device, rather than a proxy devised by a malicious SD. Besides, the speed of decrypting the IP core using EC-based crypto will be a bottleneck of IP configuration. As an alternative, a symmetric decryption module is added for IP

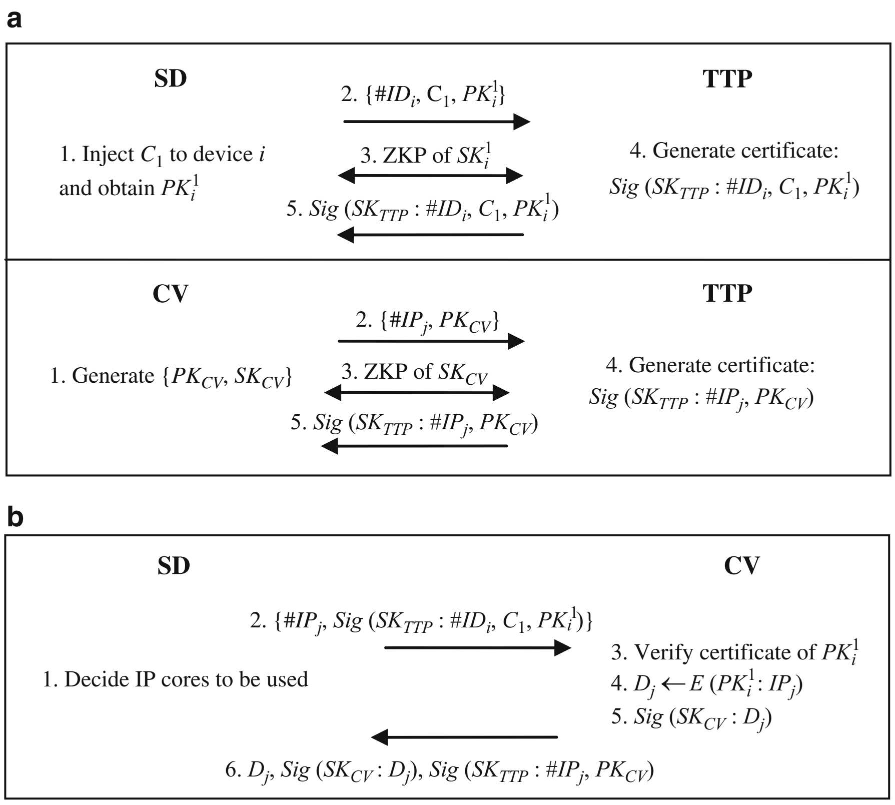

a
SD
TTP
2. {#ID_i, C_1, PK_i^1}
1. Inject C_1 to device i and obtain PK_i^1
3. ZKP of SK_i^1
4. Generate certificate: Sig (SK_{TTP} : #ID_i, C_1, PK_i^1)
5. Sig (SK_{TTP} : #ID_i, C_1, PK_i^1)
CV
TTP
2. {#IP_j, PK_{CV}}
1. Generate {PK_{CV}, SK_{CV}}
3. ZKP of SK_{CV}
4. Generate certificate: Sig (SK_{TTP} : #IP_j, PK_{CV})
5. Sig (SK_{TTP} : #IP_j, PK_{CV})
b
SD
CV
2. {#IP_j, Sig (SK_{TTP} : #ID_i, C_1, PK_i^1)}
3. Verify certificate of PK_i^1
1. Decide IP cores to be used
4. $D_j \leftarrow E$ (PK_i^1 : IP_j)
5. Sig (SK_{CV} : D_j)
6. D_j, Sig (SK_{CV} : D_j), Sig (SK_{TTP} : #IP_j, PK_{CV})

Fig. 10 Public-key crypto and strong PUF based FPGA IP licensing protocol [44]. (**a**) Public key enrollment. (**b**) IP core licensing

decryption so that the EC-based crypto core is used only for the establishment of the secret IP decryption key. This proposal results in a protocol similar to the one shown in Fig. 9 for IP decryption.

Of particular importance is a common assumption of the PUF based licensing protocols described above: during the IP licensing stage and thereafter, the PUF response R is only available internally to the on-chip decryption and authentication circuit. As explicitly stated in [38, 44], after the enrollment stage, the circuit used to obtain the CRPs is assumed to be destroyed, e.g., by blowing fuses. For security reason, a CRP is usually used only once in the IP licensing scheme. The recorded CRP list must be long enough to last for the lifetime of the device. This imposes an exorbitant storage requirement on the TTP [36]. Recent developments in strong PUF with a secret model [52] may help to alleviate the storage requirement. With the secret model, which is essentially the characterization information of a PUF, the TTP can simulate and obtain the response of an arbitrary challenge to the PUF. By saving the characterization information and associated parameters of the PUF instead of its many CRPs, the storage space can be cut down tremendously.

Recent research results on PUFs may subvert the FPGA IP licensing schemes described above. For example, numerous strong PUFs with a public model, denoted as PPUF, have been proposed [53–57]. A typical form of PPUF exploits the time gap of generating the response to a challenge between the direct execution with PUF and the simulation with public model. A radically different communication protocol for secure IP transaction may emerge as any party can now obtain the CRP response using the publicly published simulation model. The secret key exchange protocol [53] and public-key communication protocol [57] have already been revamped based on the PPUF. If the licensing scheme of FPGA IP cores can leverage on such PPUF based protocols, there will be no need to have authenticated and secure communication channels and the secure key establishment can be achieved without the high implementation cost of conventional public-key crypto such as ECC and RSA.

6 Conclusion

A suitable licensing model for FPGA based IP cores, where the licensed IP cores are securely protected, is vital for the FPGA IP market to thrive. Compared to the upfront IP licensing model currently used in the market, pay-per-use IP licensing has two attractive advantages. A major group of consumers of the FPGAs and the IP buyers will benefit from the competitive pricing when the cost of IP usage is commensurate with the number of end products sold. By proportionally charging the use of the IP core to the sale of the system developed, the risk for the system developers will be considerably reduced and the core developers will also gain from a more vibrant FPGA market with greater demand for more sophisticated and versatile IP cores. A comprehensive overview of the methods realizing such a new licensing model is provided in this chapter. These methods are dichotomized into conventional crypto primitives based schemes and the PUF based schemes. The former requires typically a secret key to be stored in the on-chip secret NVM, while the latter uses the PUF response as the device secret key. Both types of methods require the use of either a combination of TTP and symmetric-key crypto or an asymmetric-key crypto to establish the device specific key for secure core installation, which is a critical element for realizing usage based licensing model.

The various attacks to FPGA IP cores and the desiderata for a good IP licensing scheme are also discussed. In particular, the IP core should be protected against low-to-medium cost attacks such as IP cloning, reverse engineering, malicious tampering, and common physical and side-channel attacks. Also, the implementation cost of the licensing scheme needs to be kept low to be receptive. By virtue of its unclonability and tamper-resistance, PUF offers greater assurance in unique device identification and secrecy in the exchanges of confidential information. The application of two types of PUF primitives, i.e., the weak PUF and the strong PUF, in the licensing protocol are discussed. The weak PUF is essentially used to generate a device specific on-chip secret key and the strong PUF provides a large CRP space to facilitate the protection of different IP cores with simplified communications. With the emergence of new PUF primitives with exquisite features, such as PPUF that exploits the execution and simulation time gaps to realize the public-key equivalent cryptography, fundamentally different secure and cost-effective IP licensing schemes from the schemes discussed in this chapter may be constructed. It is hoped that the discussions will shed some light on the promising solutions to advocate the impending pay-per-use licensing model for FPGA IPs.

References

1. Maxfield, C.: Xilinx announces world's highest capacity FPGA. EE Times (2011). http://www.eetimes.com/document.asp?doc_id=1260468
2. Maxfield, C.: The Design Warrior's Guide to FPGAs: Devices, Tools and Flows. Elsevier, MA (2004). ISBN: 978-0-7506-7604-5
3. Xilinx: AXI4 Interconnect Paves the Way to Plug-and-Play IP (v1.0). White paper 379 (2010)
4. Xylon: Xylon Low-Volume IP Program (v1.00). Application Note 0022 (2010)
5. Transparency Market Research: Field-Programmable Gate Array (FPGA) Market – Global Industry Analysis, Size, Share, Growth, Trends and Forecast, 2013–2019 (2013)
6. Lattice Semiconductor: Third Generation Non-volatile FPGAs Enable System on Chip Functionality. White paper (2007)
7. Altera: An FPGA Design Security Solution Using a Secure Memory Device (v1.0). White Paper 01033 (2007)
8. Xilinx: FPGA IFF Copy Protection Using Dallas Semiconductor/Maxim DS2432 Secure EEPROMS (v1.1). Application Note 780 (2010)
9. Xilinx: Security Solutions Using Spartan-3 Generation FPGAs (v1.1). White Paper 266 (2008)
10. McNeil, S.: Solving Today's Design Security Concerns (v1.2). Xilinx White Paper 365 (2012)
11. Kean, T.: Secure configuration of a Field Programmable Gate Array. In: IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 259–260 (2001)
12. Bossuet, L., Gogniat, G., Burleson, W.: Dynamically configurable security for SRAM FPGA bitstreams. Int. J. Eng. Sci. **2**(1/2), 73–85 (2006)
13. Guneysu, T., Moller, B., Paar, C.: Dynamic intellectual property protection for reconfigurable devices. In: International Conference on Field-Programmable Technology, pp. 169–176 (2007)
14. Jain, A.K., Yuan, L., Pari, P.R., Qu, G.: Zero overhead watermarking technique for FPGA designs. In: ACM Great Lakes symposium on VLSI, Washington, DC, pp. 147–152 (2003)
15. Ziener, D., Teich, J.: Power signature watermarking of IP cores for FPGAs. J. Signal Process. Syst. Signal Image Video Technol. **51**(1), 123–136 (2008)
16. Liang, W., Sun, X.M., Xia, Z.H., Sun, D.C., Long, J.: A chaotic IP watermarking in physical layout level based on FPGA. Radioengineering **20**(1), 118–125 (2011)
17. Lach, J., Mangione-Smith, W.H., Potkonjak, M.: Fingerprinting techniques for field-programmable gate array intellectual property protection. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. (TCAD) **20**(10), 1253–1261 (2001)
18. Caldwell, A.E., Choi, H.J., Kahng, A.B., Mantik, S., Potkonjak, M., Qu, G., Wong, J.L.: Effective iterative techniques for fingerprinting design IP. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **23**(2), 208–215 (2004)
19. Chang, C.H., Zhang, L.: A blind dynamic fingerprinting technique for sequential circuit intellectual property protection. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **33**(1), 76–89 (2014)
20. Santarini, M.: Xilinx ships industry's first 20-nm all programmable devices (2014). http://www.xilinx.com/support/documentation/xcell_articles/xcell86-xilinx-ships-industry-first-20nm.pdf

21. Mouli, C., Carriker, W.: Future fab: how software is helping Intel go nano-and beyond. IEEE Spectr. **44**(3), 38–43 (2007)
22. Note, J.-B., Rannaud, E.: From the bitstream to the netlist. In: International ACM/SIGDA Symposium on Field Programmable Gate Arrays, Monterey, pp. 264–264 (2008)
23. Benz, F., Seffrin, A., Huss, S.A.: Bil: a tool-chain for bitstream reverse-engineering. In: International Conference on Field Programmable Logic and Applications (FPL), Oslo, pp. 735–738 (2012)
24. Lattice Semiconductor: FPGA Design Security Issues: Using the ispXPGA Family of FPGAs to Achieve High Design Security, White paper (2003)
25. Tehranipoor, M., Koushanfar, F.: A survey of hardware Trojan taxonomy and detection. IEEE Des. Test Comput. **27**(1), 10–25 (2010)
26. Khelil, F., Hamdi, M., Guilley, S., Danger, J.L., Selmane, N.: Fault analysis attack on an FPGA AES implementation. In: New Technologies, Mobility and Security (NTMS), pp. 1–5 (2008)
27. Saha, D., Mukhopadhyay, D., Roy Chowdhury, D.: A diagonal fault attack on the advanced encryption standard. ICAR Cryptology ePrint Archive, 581 (2009)
28. Kocher, P.: Timing attacks on implementations of Diffie-Hellman, RAS, DSS, and other systems. In: Advances in Cryptology-CRYPTO, pp. 104–113. Springer, Berlin (1996)
29. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology, pp. 388–397. Springer, Berlin (1999)
30. De Mulder, E., Buysschaert, P., Ors, S.B., Delmotte, P., Preneel, B., Vandenbosch, G., Verbauwhede, I.: Electromagnetic analysis attack on an FPGA implementation of an elliptic curve cryptosystem. In: International Conference on Computer as a Tool, pp. 1879–1882 (2005)
31. Abraham, D.G., Dolan, G.M., Double, G.P., Stevens, J.V.: Transaction security system. IBM Syst. J. **30**(2), 206–229 (1991)
32. NIST: FIPS 197: Advanced Encryption Standard (AES) (2001)
33. Stigge, M., Plötz, H., Müller, W., Redlich, J.-P.: Reversing CRC – Theory and Practice. Technical Report SAR-PR-2006-05, Humboldt University Berlin (2006)
34. NIST: FIPS 180-3: Secure Hash Standard (SHS) (2008)
35. Peterson, E.: Developing Tamper Resistant Designs with Xilinx Vertex-6 and 7 Series FPGAs. Xilinx Application Note 1084 (2012)
36. Ruhrmair, U., Devadas, S., Koushanfar, F.: Security based on physical unclonability and disorder. In: Tehranipoor, M., Wang, C. (eds.) Introduction to Hardware Security and Trust. Springer, Berlin (2011)
37. MicroSemi: SmartFusion2 Soc FPGA Reliability and Security User Guide (Rev. 3) (2013)
38. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Cryptographic Hardware and Embedded Systems (CHES), pp. 63–80 (2007)
39. Kean, T.: Cryptographic rights management of FPGA intellectual property cores. In: ACM/SIGDA Symposium on Field Programmable Gate Arrays (FPGA), pp. 113–118 (2002)
40. Drimer, S., Guneysu, T., Kuhn, M.G., Paar, C.: Protecting Multiple Cores in a Single FPGA design (2008). http://www.saardrimer.com/sd410/papers/protect_many_cores.pdf
41. Maes, R., Schellekens, D., Verbauwhede, I.: A pay-per-use licensing scheme for hardware IP cores in recent SRAM-based FPGAs. IEEE Trans. Inf. Forensics Secur. **7**(1), 98–108 (2012)
42. Zhang, L., Chang, C.H.: A pragmatic per-device licensing scheme for hardware IP cores on SRAM based FPGAs. IEEE Trans. Inf. Forensics Secur. **9**(11), 1893–1905 (2014)
43. Simpson, E., Schaumont, P.: Offline hardware/software authentication for reconfigurable platforms. In: Cryptographic Hardware and Embedded Systems (CHES), pp. 311–323 (2006)
44. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: Physical unclonable functions and public-Key crypto for FPGA IP protection. In: International Conference on Field Programmable Logic and Applications, pp. 189–195 (2007)
45. Kean, T.: Method of protecting intellectual property cores on field programmable gate array. US Patent Application 10/172,802; Publication number: US20020199110 A1 (2002)
46. Gaspar, L., Fischer, V., Guneysu, T., Cherif Jouini, Z.: Two IP protection schemes for multi-FPGA systems. In: International Conference on Reconfigurable Computing and FPGAs, Cancun, pp. 1–6 (2012)

47. Drimer, S.: Authentication of FPGA bitstreams: why and how. In: International Conference on Reconfigurable Computing: Architectures, Tools and Applications, pp. 73–84 (2007)
48. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: Keccak Implementation Overview (version 3.2) (2012). http://keccak.noekeon.org/Keccak-implementation-3.2.pdf
49. Regazzoni, F., Wang, Y., Standaert, F.-X.: FPGA implementations of the AES masked against power analysis attacks. In: International Workshop on Constructive Side-Channel Analysis and Secure Design, Darmstadt, pp. 56–66 (2011)
50. McEvoy, R., Tunstall, M., Murphy, C.C., Marnane, W.P.: Differential power analysis of HMAC based on SHA-2, and countermeasures. In: International Conference on Information Security Applications, Jeju Island, pp. 317–332 (2007)
51. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J. ACM **38**(3), 691–729 (1991)
52. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Techniques for design and implementation of secure reconfigurable PUFs. ACM Trans. Reconfigurable Technol. Syst. **2**(1), 1–33 (2009). Article 5
53. Beckmann, N., Potkonjak, M.: Hardware-based public-key cryptography with public physically unclonable functions. In: International Workshop on Information Hiding (IH), Darmstadt, pp. 206–220 (2009)
54. Potkonjak, M., Meguerdichian, S., Nahapetian, A., Wei, S.: Differential public, physically unclonable functions: architecture and applications. In: ACM/EDAC/IEEE Design Automation Conference (DAC), New York, pp. 242–247 (2011)
55. Meguerdichian, S., Potkonjak, M.: Matched public PUF: ultra low energy security platform. In: International Symposium on Low Power Electronics and Design (ISLPED), pp. 45–50 (2011)
56. Meguerdichian, S., Potkonjak, M.: Using standardized quantization for multi-party PPUF matching: foundations and applications. In: IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 577–584 (2012)
57. Teng, X., Wendt, J.B., Potkonjak, M.: Digital bimodal function: an ultra-low energy security primitive. In: IEEE International Symposium on Low Power Electronics and Design (ISLPED), pp. 292–296 (2013)

Part III
Trust in Softwares, Networks and Services

Heterogeneous Architectures: Malware and Countermeasures

Flavio Lombardi and Roberto Di Pietro

Abstract Malware is becoming smarter and stealthier and it is increasingly widespread over a large number of heterogeneous platforms. Most often, malicious software is especially built for a given target environment as it leverages its peculiarities. However, many similarities exist among malicious approaches. Such common features can be used to prevent, detect and react to such threat. This chapter discusses the above introduced threat and shows how advanced virtualization technology (quite common on most multicore CPU architectures) can be of help in monitoring, analyzing and protecting heterogeneous software/hardware architectures from malware.

1 Introduction

New malware spreads daily over the Internet, affecting services and users everywhere. Malicious software is increasingly widespread on a large number of heterogeneous platforms (Server/Cloud/PC/Mobile) and operating systems (Windows, OS X, Linux, Android, ...) [67]. In addition, malware is becoming more complex and stealthy by making use of obfuscation and condition-based triggering [6, 31, 35]. Such malware can affect the reliability of the systems, gather sensitive data and gain access or attack larger computing systems [36]. Due to the pervasiveness of computing devices, viruses, rootkits, worms, botnets, keyloggers and other types of malware have recently broadened their focus from desktop PCs and servers to smartphones and also implantable medical devices (IMD), where they can reach a large number of different targets (see also BYOD security issues [71]).

F. Lombardi (✉)
IAC-CNR, Rome, Italy
e-mail: flavio.lombardi@cnr.it

R.D. Pietro
Mathematics and Physics Department, Roma Tre University, Rome, Italy
e-mail: dpietro@mat.uniroma3.it

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_13

As a matter of fact, malicious software implementations differ among platforms. However, similarities exist among them as regards malware behavior and attack strategies. To defend against recent complex threats, more sophisticated analysis, detection and prevention techniques are needed.

In fact, as malware becomes stealthier, its detection becomes a difficult and time-consuming process comprising in-depth monitoring of code, identification and analysis of suspicious components. Two main popular anti-malware approaches exist: Static Analysis and Dynamic Analysis [17]: the former approach analyzes malicious code by decompiling the application in order to access its binary or even reconstruct source code; in dynamic analysis, possibly malicious code is deployed into a contained environment where behavior and interactions with the external world is monitored, recorded, and analyzed.

Both these two approaches have advantages and pitfalls, as we will see in the following sections. It is however important to stress that static and dynamic analysis may require large computing and storage resources as the amount of code to be analyzed can be large and the analysis of its behavior can be quite complex. In particular, dynamic analysis is oftentimes particularly time consuming as malware actions can be triggered by rare events that activate malicious behavior only when certain conditions are met [31]. Furthermore, the amount of collected data can be quite large. It is clear that an evolution of the detection tools is needed in order to cope with the ever increasing number of mobile applications and toolkits that are introduced or updated every day.

The aim of this chapter is to help creating state-of-the-art research references supporting both practitioners and researchers in building malware-resilient systems. One of the main technologies that will be adopted to achieve this goal is hardware virtualization. Further, the malware obfuscation problem is introduced as well as effective static and dynamic code analysis approaches and tools. In particular, novel algorithms for code execution path tracing, effective storage and analysis are surveyed and best practices are suggested.

In particular, this chapter is organized as follows: in the Introduction we have introduced the malicious software threat and has motivated the rest of the chapter. In the Technology Background section we will discuss existing malware over a range of platforms such as Windows, Mac OS X, iOS, Desktop/Server Linux and Android. This section will also highlight the similarities and differences among them. It will then discuss malware detection and reaction approaches and tools over different platforms (Server, Desktop, Mobile) and Operating System (Proprietary and Open Source). Further, it will show virtualization technology benefits and limitations on both x86 and ARM CPUs. The Additional Considerations section will introduce the reader to some fundamental examples of static and dynamic approaches. The Static Analysis section will introduce the main results of Static Analysis approaches. The Dynamic Analysis section will introduce the main results of Dynamic Analysis approaches. The section titled Practical Hints for Practitioners will show the practitioner how to leverage latest technology and approaches making use of manycore and virtualization technology for introspection and malware detection and analysis. The section on Future Research Directions will highlight the future trend of Secure Computing and of malware resilience and BYOD security in complex distributed scenarios such as (mobile) Cloud Computing. The Conclusion section will summarize the information discussed in the previous sections and introduces the reader to new perspectives for transparently and securely detecting and removing malware from heterogeneous platforms. Finally, the bibliography is reported, containing the most relevant works cited in this chapter, in order for the reader to be able to further delve into this exciting and relevant domain.

2 Technology Background

Present and future technological landscapes are particularly interesting, both for an attacker and for the administrator of the systems that have to be protected. Virtualization technology has a long history of successes, ranging from the initial idea IBM devised for mainframe task isolation [12] to the latest hypervisor-based technology that is the enabling factor for present cloud services. In particular hardware-based virtualization has introduced effective isolation and monitoring benefits in Virtual Machines deployed all over the cloud [38]. This is especially true on x86 CPUs [51] (see Fig. 1) where the additional super-privileged execution layer is protected by hardware mechanisms. Similarly, recent ARM CPUs added support for additional super-privileged execution rings. As such, hardware-supported virtualization is starting to be commonplace also for the ARM platforms [13].

Virtualization has engaged a never-ending battle both on the attacker side and on the OS side. As an example of the former, the BluePill [57] malware uses virtualization to trick the user into a controlled fake environment where his activity and data can be eavesdropped. As examples of the latter, virtual machines run both enterprise desktop and server OSes, leveraging isolation benefits. Virtualization also introduces technological challenges that deserve special attention. Among them, an increased complexity of digital forensics [52] and issues regarding novel vulnerabilities of the host system [57].

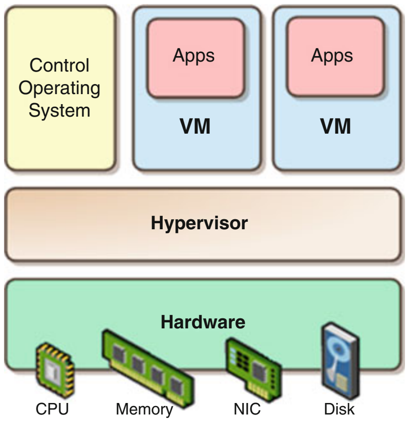


Fig. 1 Virtual machines and execution rings [54]

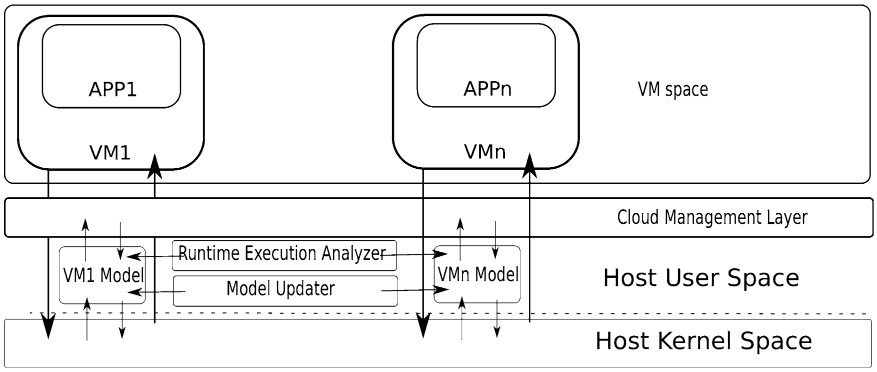


Fig. 2 Example of an advanced virtualization-supported monitoring architecture

However, container-based virtualization [18] has recently emerged as a more lightweight virtualization alternative with respect to the hardware-supported one. The benefits of reduced code size due to sharing the OS kernel among different containers (i.e. self-managed application boundaries) comes at the expense of a greater complexity of the deployed instances. Further, containers feature a thinner isolation layer, as the OS kernel is the very same across different instances. As such, the reduced protection from sibling-tasks attacks of container-based virtualization approaches has to be taken into account by system administrators.

In fact, the adoption of advanced hardware virtualization technology (nowadays available on most CPU architectures) is a necessary ingredient to achieve detection, analysis and protection from malware on a large number of software/hardware architectures. Hardware virtualization enables transparent execution monitoring and anomaly detection of OS and applications [16] (see Fig. 2). These characteristics are essential for protecting against security threats. In addition, effective transparency of the monitoring architecture is needed since malware could behave differently and hide its malicious behavior when aware that a monitoring service is active [41].

In the following, we survey malware status and main approaches on the most widespread platforms, starting from Microsoft Windows.

2.1 Windows Malware

The vast majority of PCs in the world run Microsoft Windows and MS Office. Vulnerabilities in Windows operating systems and their applications have been targeted by a large number of attacks in the past [46]. Protection mechanisms such as intrusion detection systems, antiviruses and rootkit scanners have constantly been improving over the years but the number of different threats is overwhelming [48]. Security updates are spread quite often by Microsoft but there is always a delay between vulnerability discovery and distribution of patches that can be

Table 1 Some example of Windows malware and possible remedies/ countermeasures

Attack type	Activity type	Possible remedy
AFX [1]	Hooks native Win API	Monitor API integrity
Conficker [62]	Alters registry keys	Sanity check on registry
HackerDefender [22]	SSDT replacement	lock SSDT from changes
Mebroot [4]	Alters disk MBR	Check MBR integrity
Rustock [70]	Alters Sysenter handler	Integrity checks
Vanquish [1]	DLL injection	library checksums

exploited by zero-day attacks. Furthermore, a very large number of PCs, (implanted) medical devices and Point of Sale (POS) still runs Windows XP, for which security update support ended on April 2014 [45]. It is worth noting that some vulnerabilities in presently available Windows OSes are backward-compatible with Windows XP and previous ones. Unfortunately, patches are only sent by Microsoft to protect supported Windows versions. POSes and implantable medical devices are particularly exposed [24]. As an example the Dexter malware [60, 63] was designed specifically to attack POS systems and steal valuable data. Similarly to other OSes, Windows is also victim of ransomware such as Cryptolocker. Such malware can show fake messages telling the user that he has been caught accessing illicit content or they completely block device functionality. The payment of the fine would then unblock the functionality and allow the user to get rid of the fake messages (Table 1).

As regards virtualized Windows OSes, they are subject to the same attacks as their bare metal counterparts, plus additional vulnerabilities that are proper of the hypervisor. Many attempts to protect the Windows kernel and its services have been presented [17, 42]. However, given the complexity of the OS and the lack of knowledge over the inner parts of the kernel and of implemented services, as well as the well-known existence of NSA-backed backdoors [78], it is quite hard to evaluate the actual security/reliability of Windows OS and applications.

2.2 *Linux Malware*

Linux is usually considered much less affected by malware than other OSes. The truth is that Linux rootkits and web-based malwares are quite widespread but few people are aware of them. In particular, during 2014, a couple of serious bugs/security issues were discovered on the Linux platform, including the Shellshock Bash shell vulnerability [11] and the SSL protocol Heartbleed bug [30]. In fact, even though Linux is present only on a minority of desktop PCs out there, it is by far the most widespread server platform [26] (not to mention mobile phones). This is the reason why Linux-based servers have become the most common targets for web server attacks [23]. In particular, PHP code running on Linux servers often hides malicious code that can be used by a botnet [56, 63] to perform a DDoS attack on heterogeneous platforms over the Internet. In addition, an attacker can manage to install Linux-targeted rootkits on target machines, be them on real HW or virtual (this latter case being more and more common). In the former case, the infection can be made more resilient to reboots and software reconfigurations. In the latter case, as discussed in [16], once exploited a vulnerability on a VM, malware can easily spread out to sibling ones. This is the reason why malware on a Virtual Machine can also try and attack the host through hypervisor (mostly Linux-based) vulnerabilities. As a matter of fact, VM's effective containment and smart monitoring are a strongly needed by Linux-based virtualization hosting.

2.3 Mac Malware

Also Macs had a reputation of being immune to malware. In fact, this is no longer true [50]. In the past, various specific malicious software targeted Apple most common applications (MS Office) and platforms. As an example, Mac Defender [61] was a bogus security software that already induced many Mac users to unwanted purchases. The trojan horse spread over a large number of Macs using links with malicious content. After a large number of alert messages the user was forced to pay for a remote solution to the issue. This malware does not need admin privileges to spread into the system. Further, it can also prevent the user from downloading a corrective patch from Apple.

Mac OS X malware continues to evolve, leveraging vulnerabilities in MS Office and Java and evading Apple's Gatekeeper protection mechanisms. Gatekeeper is a tool for checking authenticity and integrity of OS X software as signed with Apple Developer ID. Unfortunately, even Apple itself was internally compromised by using a zero-day Java vulnerability [63].

Mac platform are rarely virtualized as Mac as a server platform has never been particularly successful. As such vulnerabilities for virtualized Macs are not a real problem.

2.4 Web Malware and Kits

Web-hosted Malware is getting more sophisticated and hidden and can attack vulnerable web browsers using malicious client-side scripts (mostly JavaScript and Python). Cross-site scripting (XSS) uses known vulnerabilities in web-based applications and hides malware into apparently legitimate content that appears at the client web browser as if coming from a trusted site. Given that any HTML documents can mix control statements, content and formatting, markup injection can be done using user-supplied data in the resulting page. However, persistent XSS malware is even more dangerous as it is triggered when the data provided by the attacker is saved by the server. This way the malicious part is delivered in all regular pages returned to all web browser users. In addition, Flash code can also redirect the web surfer to malicious content. This way Flash users can be infected without even being redirected. In addition, malware can hide in specific video codecs.

As an example, Darkleech [23, 63] attacks are particularly dangerous since they hide and get active with irregular patterns. Darkleech is smart as it maintains IP blacklists to avoid repeating a malicious activity twice on the same machine.

As regards software kits for customizing malware to target specific organizations and users, Zeus [55] is one of the most notable representatives. Another relevant example is Blackhole [5] exploiting Java, Flash and PDF.

Some attacks are targeted at specific companies, victims of advanced persistent threat (APT) attacks [10, 47]. In this case, malware behaves as a legitimate application and is often signed with the software producer's key. Unfortunately, at given times such malware executes malicious activity. APTs can go undetected for months. As an example, Blame [63] accesses malicious content from a modified version of an open source multimedia library. As such, in theory, it should be easier to check code integrity. Another example is Simbot [69], which distributes a clean but vulnerable application that contains a very long command line that hides a shellcode. Such shellcode then loads the main payload.

Redkit [63]is another very smart malware targeting websites and redirecting users from a legitimate site to another legitimate site that has been compromised before.

2.5 Mobile Malware

Even though relevant improvements on malware detection for desktop and server PCs have been made, leveraging such techniques to mobile devices is quite challenging. Attacks targeting mobile devices are increasing in quantity and quality. Existing mobile malware detection techniques are similar to traditional desktop solutions. However, they are not effective due to the limited computing resources and battery constraints of mobile devices [66]. It is worth noting that mobile devices are often equipped with powerful sensing capabilities that can collect a very large variety of data from the surrounding environment [15]. Voice, images, temperature, humidity, pressure, location, wireless networks and many other data can be collected by those devices. Mobile devices can also interact with the environment via infrared, bluetooth, light, NFC and WiFi channels. They have also the ability to incorporate third-party apps from different markets. These powerful capabilities pose strong security and privacy issues to users, particularly through malicious software that can get access to the data collected by the device and can interact with the external world in an almost unlimited number of ways. Mobile platforms provide mechanisms to help prevent malicious apps to be installed on devices. The most successful approach is that of vetting applications before they can be available for installation. However, such a priori static analysis is not fully effective as malicious content can get to the device in a number of ways even when the app appears non-malicious (similarly to the web-based attacks shown above).

As suggested in the following sections, a comprehensive protection approach has to be devised leveraging recent technological advances of these platforms. In particular. as ARM virtualization support increases, monitoring and containment techniques borrowed from PCs and servers can be deployed [38].

3 Practical Hints for Practitioners

Even though a complete solution for the security and privacy issues on heterogeneous platforms has not yet been devised, some rules can be effective in reducing the risks associated with present malware threats.

OS and application software have to be continuously patched with up-to-date security fixes. Old attacks are sometimes still effective due to the lack of updates. Further, if the OS supports limiting the installable apps to certified apps from the official Store, this option further reduces risks.

It is better to disable or remove Java from the client system if this choice is available [20]. A large number of malicious software is particularly focused on Java. Java is especially vulnerable inside web browsers. That is where they see the greatest vulnerabilities are at the moment. Further, in general, it is better to remove unnecessary browser plugins.

3.1 Web Remedies

Web server attacks are quite difficult to detect. However, some effective approaches for protecting servers and clients can be based on layered protection [74]. Web filtering and dynamic analysis with runtime detection and IDSes can be jointly used [41].

3.2 Mobile Remedies

Felt [53] surveyed mobile malware and gave a taxonomy that is based on actual specimens. Also Tangil and Sadeghi [58, 66] actively report on mobile malware on the Android platform. Most results are based on specific malware specimens that have been isolated and are now available to registered experts in the field. In particular, a rich set of mobile malware is hosted by [79]. Chandramohan [9] proposed a tool for effective identification of android malware.

VirusMeter, a general malware detection method proposed by Liu [37] detects anomalous behavior on mobile devices. The rationale underlying VirusMeter is the fact that mobile devices are usually battery powered and any malicious activity would inevitably consume some battery power. By monitoring power consumption on a mobile device, VirusMeter catches misbehaviors that lead to abnormal power consumption. For this purpose, VirusMeter relies on a concise user-centric power model that characterizes power consumption of common user behaviors. In a real-time mode, VirusMeter can perform fast malware detection with trivial runtime overhead, in battery-charging mode, malware detection can be devoted more resources and thus it can be speeded-up.

4 Malware Detection and Analysis

The problem of detecting and analyzing malware has received considerable attention over the past few years. More in general, finding software errors could be in a way similar to discovering vulnerabilities. Several approaches in the literature use both White Box and Black Box analysis approaches, aiming at spotting software errors and malicious behavior [65, 73]. It is true that decompilers are strong enablers for white box analysis. However, obfuscation techniques are increasingly sophisticated and render White Box testing unfeasible. As a consequence, Black Box testing is generally used when source code is not available. However, both approaches study applications using as input a simulation of users' behavior.

We now discuss the most relevant research contributions on the two main approaches to software analysis that are related to White Box and Black Box, namely Static and Dynamic analysis.

4.1 Static Analysis

The malware analysis process is split into static analysis and dynamic analysis. Both static and dynamic analysis have their own strengths and weaknesses. As an example of pure static analysis, Kang [31] uses binary content comparison to classify malware. Slicingdroids [25] uses a static Android analysis framework to analyze smali code and to create program slices in order to perform data-flow analyses to backtrack parameters used by a given method. This helps to identify suspicious code regions in an automated way. Some tools are aimed at extracting static features of malware, such as Pingali [77], that focuses on ripping malicious features from Microsoft Windows binaries.

Calvet et al. [8] look for easily identifiable static features of cryptographic functions in obfuscated code. They present a tool that leverages this fact to identify cryptographic functions in obfuscated programs, by retrieving their I/O parameters in an implementation-independent fashion, and comparing them with those of known cryptographic functions. However, Schrittwieser et al.[59] demonstrate the incompleteness of these models leveraging on the gap between model and machine to stay undetectable. They introduce "covert computation", which implements functionality in side effects of microprocessors where the flags register can be used to compute basic arithmetical and logical operations. Lee [34] uses static analysis to calculate the similarity between two files to be executed through by comparing character strings to identify and classify malware.

4.2 Code Obfuscation

One of the most relevant issues with static analysis is obfuscated code. As a matter of fact, applying static analysis techniques in such scenarios does not produce meaningful results. However, other approaches are viable such as those proposed by Ker et al. [32] describing most relevant steganography and steganalisys techniques including active [21] and passive wardens. While cryptographic techniques for malware obfuscation are well described by Apriville [2], a possible solution lies with software tainting approaches that induce and observe behavioral changes in the software [65]. This is one of the cases where dynamic analysis has advantages over static analysis, as detailed in the following.

4.3 Dynamic Analysis

Malware has been classified in various ways using dynamic analysis. As an example, CAMAS [44] is a tool for the analysis and classification of malicious Android applications, through pattern recognition on execution graphs. The framework analyzes behaviors at system-call level and exploits the concept of Action Node to store relationships between actions. The framework finds common subgraphs in malware executions and classifies other apps by searching for common patterns of the previously mined subgraphs. Execution Path Analisis (EPA) [16, 19, 27] is one of the most promising approaches that leverage past execution traces to learn about and possibly prevent malicious actions to take place. In particular, virtual execution environments are used for malware analysis [76] this can be of help in collecting traces and data. Virtualization is often used in combination with EPA. However, virtualization can be detected and it can interfere with malware behavior.

Further, in order to better evaluate software behaviour, model checking can be used [14, 72]. A notable example is Crowdroid [7], i.e. a machine learning-based malware detection solution that builds a vector of n features (the Android system calls) and it works by analyzing the number of times each system call has been issued by an application during execution. The rationale behind is that a genuine application differs from its malicious version, since it issues different types and a different number of system calls.

Analysis of Causal Execution Differences is also relevant in dynamic analysis. Johnson [29] proposes a differential slicing approach that automates the analysis of execution differences. Differential slicing outputs a causal difference graph that captures the input differences that triggered the observed difference and the causal path of differences that led from those input differences to the observed difference. The analyst can then use the graph to understand the observed difference. Johnson can successfully identify the input differences that caused the observed difference. Such causal difference graph significantly reduces the amount of time and effort required for an analyst to understand the observed difference.

Differential Fault Analisis [65] is a dynamic analysis technique that aims to bridge the semantic gap and better classify code behavior by establishing cause-effect relationships between changes induced by the analyst and the visible effects they produce. This approaches offer advantages when the code to analyze is obfuscated, albeit it can have higher complexity and computing costs.

5 Future Research Directions

Businesses increasingly rely on cloud-hosted services for managing internal and external data. This exposes both personal and corporate data.

BYOD issues are becoming of primary importance, as such devices can help rendering corporate protection mechanisms useless. The BYOD security risk can be mitigated by enforcing stricter security policies on such devices when inside corporate borders. This includes preventing side-loading of mobile apps from unknown sources and advanced anti-malware protection. However, the key enabling factor for effective security of BYOD devices is the enforcement of trusted platform architectures via lightweight trusted hypervisors [40]. Such virtual machine manager would be able to switch userspace in use based on a policy that could take into consideration location/position and kind of data being managed. This will also be effective against data exfiltration attempts. Some other interesting approaches will involve secure reputation-base ranking of services, cloud node security ranking based on past behavior.

Further, NSA-supposedly-supported backdoors and vulnerabilities such as those in the OpenSSL and Windows OSes are probably there to stay. However, improvements in the software verification tools may render it possible to analyze and improve (at least open-source) current and future implementations of relevant cryptographic protocols.

In addition, stronger authentication mechanisms and cloud data access policies are becoming increasingly important. Password-based mechanisms will eventually disappear in the face of increasingly powerful brute force attacks.

Further, latest technology and novel approaches can make use of advanced manycore CPUs and GPUs together with virtualization technology to allow a more effective execution monitoring and malware detection.

In particular, visualizing and classifying malware using image processing techniques can be a successful trend in malware analysis. As an example, in malware visualization, binaries are visualized as images. Indeed, for many malware families, the images belonging to the same family appear very similar in layout and texture. A classification can then be performed such as in [48].

Further, advanced anomaly monitoring systems will be able to verify on-the-fly whether running programs comply to their expected normal behavior or not [46], based on a rich set of discriminators, and relative entropy between distributions of system calls. A sound approach today, anomaly-based detection has a bright future ahead, especially when combined with powerful parallel manycore capabilities [64] such as those found on recent GPUs [43].

5.1 Virtualization and Introspection

As discussed above, the additional privileged layer protected by hardware virtualization features, allows guest applications and OSes to be constantly monitored and analyzed. This is the main idea behind approaches that aim at transparently collecting as much information as possible from the guest in order to be able to reconstruct a faithful and realistic model where policies are evaluated [16, 41]. The main difficulty of these approaches is that the gap between collecting guest information and being able to build a model is quite large. Semantic introspection [68] is the set of technologies that allow bridging such gap and reconstructing a faithful model.

5.2 Cloud-Based Analysis

As traditional server workloads, also malware analysis tasks can be offloaded to the cloud. For example, when power consumption is one major constraint (e.g. battery-powered devices) that makes unaffordable to run traditional detection engines, cloud-based approaches can be effective albeit rising some privacy concern.

It is also possible [49] to spend bandwidth resources to significantly reduce on-device CPU, memory, and power resources. As an example, mobile antivirus functionality can be moved to an off-device network service employing multiple virtualized malware detection engines. A cloud-based engine can reduce on-device software complexity, while allowing for platform-specific behavioral analysis engines.

Cloud-based approaches have been proposed to deal with malware complexity and the variety of malware over the Internet. They generate malware signatures for anti-virus (AV) scan engines. An automatic malware signature discovery system for AV cloud (AMSDS) [75] was created to generate malware signatures from both static and dynamic aspects. The limitation here is that signature-based analyses can be easily circumvented by obfuscated malwares. As such, anti-malware services that use pre-existing web-based file scanning services for malware detection, such as ThinAV [28] can provide some form of real-time anti-malware remote scanning.

The first results of this approaches are interesting. However, we believe the real novelty/quantum leap of cloud-based anti-malware research lies with cloud-based clones or real mobile phones. In fact, as shown by Costa [33] with Clone2Clone (C2C), a distributed peer-to-peer platform for cloud clones of smartphones is possible. C2C offloads communication between smartphones on the cloud. Device-clones hosted in virtualization environments on clouds. C2C makes it possible to implement distributed execution of advanced peer-to-peer services in a network of mobile smartphones reducing the cellular data traffic and saving the battery for respectively security checks.

Further, virtual copies of real smartphones (the clones) that run on the cloud, synchronized with the corresponding devices, can help alleviate the computational burden on the real smartphones. CloudShield [3] a suite of protocols running on P2P networks of clones, organizes clones in a P2P network in order to facilitate content sharing among smartphones. CloudShield is used to compute the best strategy to patch the smartphones in such a way that the number of devices to patch is low and that the worm is stopped quickly.

5.3 Manycore-Based Analysis

Recent CPU and GPU technology has managed to overcome Moore's law present limitations by multiplying the number of computing cores. In particular, Graphics Processing Units can be used to quickly validate and check host (i.e. CPU) code and data. This is the main idea behind CUDACS [39] and various other approaches that aim at concurrently executing code and monitoring/protecting its execution (see Fig. 3). This promising area is just in its infancy, given that only recent APUs with unified memory addressing allow a stricter integration between the two computing approaches. Thus heterogeneous redundancy allows bidirectional checking that can also speed-up static and dynamic analysis by parallelizing the workload.

6 Conclusion

In this chapter, we surveyed malicious software issues on a large number of wide-spread platforms, including Microsoft Windows, Linux, Mac and mobile platforms. The main characteristics and attack strategies of most recent malware have been described, showing similarities and peculiarities among them. Advice and guidelines on how to reduce the attack surface and limit the amount of damage such malware can do have been given.

However, as malware becomes smarter and stealthier, its detection becomes a complex process comprising monitoring, data collection and analysis. The two main popular detection approaches, Static Analysis and Dynamic Analysis, inspect malicious code or deploy such code into a contained environment. However, both

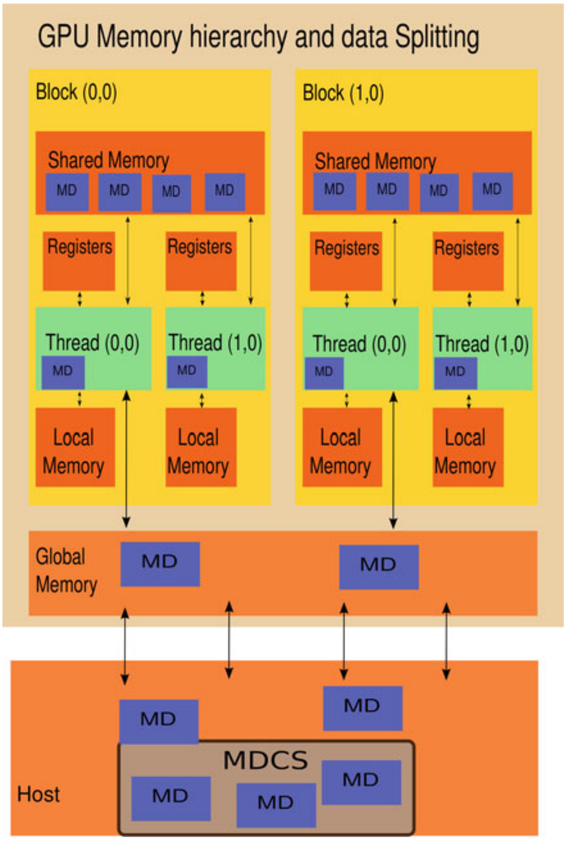


Fig. 3 Manycore-based analysis on a GPU [39]. Monitoring Data (MD) is sent to the GPU by a Monitoring Data Collection System (MDCS) on the host device

static and dynamic analysis require large computing and storage resources. An evolution of the analysis tools is needed in order to cope with the ever increasing number of malicious applications and toolkits. Novel technologies and approaches, such as General Purpose GPU computing, recently introduced, have not fully been leveraged to their full potential. As such, the future as room for evolution of both malware and anti-malware techniques and approaches, as surveyed in present chapter.

References

1. Alexander, J.S., Dean, T., Knight, S.: Spy vs. spy: Counter-intelligence methods for backtracking malicious intrusions. In: Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research, CASCON '11, pp. 1–14. IBM Corp, Riverton, (2011)
2. Apvrille, A.: Cryptography for mobile malware obfuscation. http://www.fortiguard.com/files/FNMS-305-Apvrille-Revised.pdf (2011)

3. Barbera, M., Kosta, S., Stefa, J., Hui, P., Mei, A.: Cloudshield: Efficient anti-malware smartphone patching with a p2p network on the cloud. In: 012 IEEE 12th International Conference on Peer-to-Peer Computing (P2P), 2, pp. 50–56 (2012)
4. Bianchi, A., Shoshitaishvili, Y., Kruegel, C., Vigna, G.: Blacksheep: Detecting compromised hosts in homogeneous crowds. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12, pp. 341–352. ACM, New York (2012)
5. BlackHat: Black hole. http://media.blackhat.com/bh-us-12/Briefings/Jones/BH_US_12_Jones_State_Web_Exploits_Slides.pdf, 2013
6. Brumley, D., Hartwig, C., Liang, Z., Newsome, J., Song, D., Yin, H.: Automatically identifying trigger-based behavior in malware. In: Lee, W., Wang, C., Dagon, D. (eds.) Botnet Detection, volume 36 of Advances in Information Security, pp. 65–88. Springer, New York (2008)
7. Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: behavior-based malware detection system for Android. In: Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM '11, pp. 15–26. ACM, New York (2011)
8. Calvet, J., Fernandez, J.M., Marion, J.-Y.: Aligot: cryptographic function identification in obfuscated binary programs. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12, pp. 169–182. ACM, New York (2012)
9. Chandramohan, M., Tan, H.B.K.: Detection of mobile malware in the wild. IEEE Comput. **45**(9), 65–71 (2012)
10. Command Five Pty Ltd. Advanced persistent threats: A decade in review. www.commandfive.com/papers/C5-APT-ADecadeInReview.pdf (2011)
11. Common Vulnerabilities and Exposures: Cve-2014-6271. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271 (2014)
12. Creasy, R.J.: The origin of the vm/370 time-sharing system. IBM J. Res. Dev. **25**(5), 483–490 (1981)
13. Dall, C., Nieh, J.: KVM/ARM: The design and implementation of the Linux ARM Hypervisor. In: Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '14, pp. 333–348. ACM, New York (2014)
14. Dam, M., Le Guernic, G., Lundblad, A.: Treedroid: a tree automaton based approach to enforcing data processing policies. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12, pp. 894–905. ACM, New York (2012)
15. De Cristofaro, E., Di Pietro, R.: Adversaries and countermeasures in privacy-enhanced urban sensing systems. Syst. J. IEEE **7**(2), 311–322 (2013)
16. Di Pietro, R., Lombardi, F., Signorini, M.: CloRExPa: cloud resilience via execution path analysis. Futur. Gener. Comput. Syst. **32**, 168–179 (2014)
17. Di Pietro, R., Mancini, L.V.: Intrusion Detection Systems, vol. 38 of Advances in Information Security. Springer, New York (2008)
18. Docker Inc.: Build, ship and run any app, anywhere. http://www.docker.com/ (2013)
19. Fan, W., Zhou, B., Liang, H., Yang, Y.: A novel program analysis method based on execution path correlation. In: Second International Symposium on Knowledge Acquisition and Modeling, 2009. KAM '09, vol. 2, pp. 178–181 (2009)
20. Finkle, J.: U.S. warns on Java software as security concerns escalate. Reuters, http://www.reuters.com/article/2013/01/11/us-java-security-idUSBRE90A0S320130111 (2013)
21. Fisk, G., Fisk, M., Papadopoulos, C., Neil, J.: Eliminating steganography in internet traffic with active wardens. In: Revised Papers from the 5th International Workshop on Information Hiding, IH '02, pp. 18–35. Springer, New York (2003)
22. Gates, C.: Hackerdefender: Rootkit for the masses. http://www.carnal0wnage.com/papers/rootkit_for_the_masses.pdf (2012)
23. Giron, J.: Reversing the DarkLeech exploit kit. http://www.gironsec.com/blog/2013/09/reversing-the-darkleech-exploit-kit/ (2013)
24. Gollakota, S., Hassanieh, H., Ransford, B., Katabi, D., Fu, K.: They can hear your heartbeats: Non-invasive security for implantable medical devices. SIGCOMM Comput. Commun. Rev. **41**(4), 2–13 (2011)

25. Hoffmann, J., Ussath, M., Holz, T., Spreitzenbarth, M.: Slicing Droids: program slicing for Smali code. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, pp. 1844–1851. ACM, New York (2013)
26. Hua, J., Sakurai, K.: Barrier: A lightweight hypervisor for protecting kernel integrity via memory isolation. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC '12, pp. 1470–1477. ACM, New York (2012)
27. Jang, J., Brumley, D., Venkataraman, S.: Bitshred: feature hashing malware for scalable triage and semantic analysis. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS '11, pp. 309–320. ACM, New York (2011)
28. Jarabek, C., Barrera, D., Aycock, J.: Thinav: Truly lightweight mobile cloud-based anti-malware. In: Proceedings of the 28th Annual Computer Security Applications Conference, ACSAC '12, pp. 209–218. ACM, New York (2012)
29. Johnson, N.M., Caballero, J., Chen, K.Z., McCamant, S., Poosankam, P., Reynaud, D., Song, D.: Differential slicing: Identifying causal execution differences for security applications. In: Proceedings of the 2011 IEEE Symposium on Security and Privacy, SP '11, pp. 347–362. IEEE Computer Society, Washington (2011)
30. Kamp, P.-H.: Please put openssl out of its misery. Queue **12**(3), 20:20–20:23 (2014)
31. Kang, B., Kim, T., Kwon, H., Choi, Y., Im, E.G.: Malware classification method via binary content comparison. In: Proceedings of the 2012 ACM Research in Applied Computation Symposium, RACS '12, pp. 316–321. ACM, New York (2012)
32. Ker, A.D., Bas, P., Böhme, R., Cogranne, R., Craver, S., Filler, T., Fridrich, J., Pevný, T.: Moving steganography and steganalysis from the laboratory into the real world. In: Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security, IH&#38;MMSec '13, pp. 45–58. ACM, New York (2013)
33. Kosta, S., Perta, V.C., Stefa, J., Hui, P., Mei, A.: Clone2clone (c2c): Peer-to-peer networking of smartphones on the cloud. In: Presented as Part of the 5th USENIX Workshop on Hot Topics in Cloud Computing, USENIX, Berkeley (2013)
34. Lee, J., Im, C., Jeong, H.: A study of malware detection and classification by comparing extracted strings. In: Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication, ICUIMC '11, pp. 75:1–75:4. ACM, New York (2011)
35. Liang, B., You, W., Shi, W., Liang, Z.: Detecting stealthy malware with inter-structure and imported signatures. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS '11, pp. 217–227. ACM, New York (2011)
36. Lindorfer, M., Di Federico, A., Maggi, F., Comparetti, P.M., Zanero, S.: Lines of malicious code: Insights into the malicious software industry. In: Proceedings of the 28th Annual Computer Security Applications Conference, ACSAC '12, pp. 349–358. ACM, New York (2012)
37. Liu, L., Yan, G., Zhang, X., Chen, S.: Virusmeter: Preventing your cellphone from spies. In: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, RAID '09, pp. 244–264. Springer, Berlin/Heidelberg (2009)
38. Lombardi, F., Di Pietro, R.: Kvmsec: a security extension for linux kernel virtual machines. In: SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing, pp. 2029–2034. ACM, New York (2009)
39. Lombardi, F., Di Pietro, R.: CUDACS: securing the cloud with CUDA-enabled secure virtualization. In: Proceedings of the 12th International Conference on Information and Communications Security, ICICS'10, pp. 92–106. Springer, Berlin/Heidelberg (2010)
40. Lombardi, F., Di Pietro, R.: A security management architecture for the protection of kernel virtual machines. In: 2010 IEEE 10th International Conference on Computer and Information Technology (CIT), pp. 948–953 (2010)
41. Lombardi, F., Di Pietro, R: Secure virtualization for cloud computing. J. Netw. Comput. Appl. **34**(4), 1113–1122 (2011)
42. Lombardi, F., Di Pietro, R., Soriente, C.: CReW: Cloud resilience for Windows guests through monitored virtualization. In: Proceedings of the 2010 29th IEEE Symposium on Reliable Distributed Systems, SRDS '10, pp. 338–342. IEEE Computer Society, Washington (2010)

43. Luebke, D., Harris, M., Krüger, J., Purcell, T., Govindaraju, N., Buck, I., Woolley, C., Lefohn, A.: Gpgpu: general purpose computation on graphics hardware. In: SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes, p. 33. ACM, New York (2004)
44. Martinelli, F., Saracino, A., Sgandurra, D: Classifying android malware through subgraph mining. In: Proceedings of the SETOP Workshop, SETOP'13. springer LNCS (2013)
45. Microsoft: Enterprise customers support for Windows XP has ended. http://www.microsoft.com/en-us/windows/enterprise/end-of-support.aspx.
46. Milea, N.A., Khoo, S.C., Lo, D., Pop, C.: Nort: Runtime anomaly-based monitoring of malicious behavior for windows. In: Proceedings of the Second International Conference on Runtime Verification, RV'11, pp. 115–130. Springer, Berlin/Heidelberg (2012)
47. Mustafa, T.: Malicious data leak prevention and purposeful evasion attacks: An approach to advanced persistent threat (apt) management. In: Electronics, Communications and Photonics Conference (SIECPC), 2013 Saudi International, pp. 1–5 (2013)
48. Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.S.: Malware images: Visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security, VizSec '11, pp. 4:1–4:7. ACM, New York (2011)
49. Oberheide, J., Veeraraghavan, K., Cooke, E., Flinn, J., Jahanian, F.: Virtualized in-cloud security services for mobile devices. In: Proceedings of the First Workshop on Virtualization in Mobile Computing, MobiVirt '08, pp. 31–35. ACM, New York (2008)
50. O'Donnell, A.: When malware attacks (anything but windows). IEEE Secur. Priv. **6**(3), 68–70 (2008)
51. Perez, R., van Doorn, L., Sailer, R.: Virtualization and hardware-based security. IEEE Secur. Priv. **6**(5), 24–31 (2008)
52. Pollitt, M., Nance, K., Hay, B., Dodge, R.C., Craiger, P., Burke, P., Marberry, C., Brubaker, B.: Virtualization and digital forensics: A research and education agenda. J. Digit. Forensic Pract. **2**(2), 62–73 (2008)
53. Porter Felt, A., Finifter, M., Chin, E., Hanna, S., Wagner, D.: A survey of mobile malware in the wild. In: Proceedings of the 1st ACM Workshop on Security and privacy in smartphones and mobile devices, SPSM '11, pp. 3–14. ACM, New York (2011)
54. Ribeiro, M.S.: Thoughts on information technology. http://itechthoughts.wordpress.com/2009/11/10/virtualization-basics.
55. Riccardi, M., Di Pietro, R., Palanques, M., Vila, J.A.: Titans' revenge: Detecting zeus via its own flaws. Comput. Netw. **57**(2), 422–435 (2013)
56. Rodríguez-Gómez, R.A., Maciá-Fernández, G., García-Teodoro, P.: Survey and taxonomy of botnet research through life-cycle. ACM Comput. Surv. **45**(4), 45:1–45:33 (2013)
57. Rutkowska, J.: Introducing the blue pill. http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.html, (2006)
58. Sadeghi, A.-R.: Mobile security and privacy: The quest for the mighty access control. In: Proceedings of the 18th ACM Symposium on Access Control Models and Technologies, SACMAT '13, pp. 1–2. ACM, New York (2013)
59. Schrittwieser, S., Katzenbeisser, S., Kieseberg, P., Huber, M., Leithner, M., Mulazzani, M., Weippl, E.: Covert computation: hiding code in code for obfuscation purposes. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, ASIA CCS '13, pp. 529–534. ACM, New York (2013)
60. Seculert: Dexter. http://www.seculert.com/blog/2012/12/dexter-draining-blood-out-of-point-of-sales.html, (2012)
61. SecureMac: Mac defender. http://www.securemac.com/pdf/macdefender.pdf (2013)
62. Shin, S., Gu, G.: Conficker and beyond: A large-scale empirical study. In: Proceedings of the 26th Annual Computer Security Applications Conference, ACSAC '10, pp. 151–160. ACM, New York (2010)
63. Sophos: Sophos security threat report. http://www.sophos.com/en-us/medialibrary/PDFs/other/sophos-security-threat-report-2014.pdf (2014)

64. Sridharan, S., Gupta, G., Sohi, G.S.: Adaptive, efficient, parallel execution of parallel programs. In: Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '14, pp. 169–180. ACM, New York, (2014)
65. Suarez-Tangil, G., Tapiador, J.E., Lombardi, F., Pietro, R.D.: Thwarting obfuscated malware via differential fault analysis. Computer **47**(6), 24–31 (2014)
66. Suarez-Tangil, G., Tapiador, J.E., Peris, P., Ribagorda, A.: Evolution, detection and analysis of malware for smart devices. IEEE Commun. Surv. Tutorials **16**(2), 961–987 (2014)
67. Symantec: Internet security threat report 2014. http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf.
68. Tamberi, F., Maggiari, D., Sgandurra, D., Baiardi, F.: Semantics-driven introspection in a virtual environment. In: Proceedings of the 2008 The Fourth International Conference on Information Assurance and Security, pp. 299–302. IEEE Computer Society, Washington (2008)
69. Telus Security Labs.: Simbot. http://telussecuritylabs.com/threats/show/TSL20110921-05 (2013)
70. Thonnard, O., Dacier, M.: A strategic analysis of spam botnets operations. In: Proceedings of the 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference, CEAS '11, pp. 162–171. ACM, New York (2011)
71. Van Bruggen, D., Liu, S., Kajzer, M., Striegel, A., Crowell, C.R., D'Arcy, J.: Modifying smartphone user locking behavior. In: Proceedings of the Ninth Symposium on Usable Privacy and Security, SOUPS '13, pp. 10:1–10:14. ACM, New York (2013)
72. van der Merwe, H., van der Merwe, B., Visser, W.: Verifying android applications using Java PathFinder. SIGSOFT Softw. Eng. Notes **37**(6), 1–5 (2012)
73. Wagner, D., Dean, D.: Intrusion detection via static analysis. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy, SP '01, pp. 156–161. IEEE Computer Society, Washington (2001)
74. Wailly, A., Lacoste, M., Debar, H.: Vespa: Multi-layered self-protection for cloud resources. In: Proceedings of the 9th International Conference on Autonomic Computing, ICAC '12, pp. 155–160. ACM, New York (2012)
75. Wu, L., Zhang, Y.: Automatic detection model of malware signature for anti-virus cloud computing. In: 011 IEEE/ACIS 10th International Conference on Computer and Information Science (ICIS), 2, pp. 73–75 (2011)
76. Yan, L.-K., Jayachandra, M., Zhang, M., Yin, H.: V2E: combining hardware virtualization and software emulation for transparent and extensible malware analysis. SIGPLAN Not. **47**(7), 227–238 (2012)
77. Zabidi, M., Maarof, M., Zainal, A.: Malware analysis with multiple features. In: 2012 UKSim 14th International Conference on Computer Modelling and Simulation (UKSim), pp. 231–235 (2012, March)
78. Zetter, K.: How a crypto backdoor pitted the tech world against the NSA. http://www.wired.com/2013/09/nsa-backdoor/ (2013)
79. Zhou, Y., Jiang, X.: Dissecting Android malware: Characterization and evolution. In: Proceedings of the 2012 IEEE Symposium on Security and Privacy, SP '12, pp. 95–109. IEEE Computer Society, Washington (2012)

Trusted, Heterogeneous, and Autonomic Mobile Cloud

Flavio Lombardi and Roberto Di Pietro

Abstract Offloading computing to distributed and possibly mobile nodes is increasingly popular thanks to the convenience and availability of cloud resources. However, trusted mobile computing is not presently viable due to a number of issues in both the mobile platform architectures and in the cloud service implementations. The complexity of such systems potentially exposes them to malicious and/or selfish behavior. This chapter describes the state-of-the-art research on theoretical advancements and practical implementations of trusted computing on a mobile cloud. Further, mobile distributed cloud computing security and reliability issues are introduced. Discussed solutions feature different levels of resiliency against malicious and misbehaving nodes.

1 Introduction

The cloud paradigm enables effective and convenient offloading of computing tasks to distributed, possibly mobile nodes. However, trusted offloading of tasks to mobile nodes has a number of issues in both in the mobile platform architectures and in the cloud service implementations [2]. The complexity of such systems potentially exposes to malicious and selfish behavior. In the real world computing nodes are prone to failures and/or misbehaving. However, the redundancy of such nodes can be leveraged to help guarantee correctness and availability of results.

Computationally expensive tasks that can be parallelized are more rapidly completed by distributing the computation among a large number of processors. The growth of the Internet has made it possible to include every possible kind of networked node into distributed computing. As such, especially in scenarios where participants get paid for their contribution, there is incentive for cheating participants to claim compensation for work not actually performed. Security

F. Lombardi (✉)
IAC-CNR, Rome, Italy
e-mail: flavio.lombardi@cnr.it

R. Di Pietro
Mathematics and Physics Department, Roma Tre University, Rome, Italy
e-mail: dipietro@mat.uniroma3.it

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_14

schemes have been proposed to protect against such threat as well as to limit the computational overhead introduced by such systems. Weaker schemes discourage cheating by reducing its convenience, while stronger schemes let participants prove that they have performed the assigned task [38].

The aim of this chapter is to create state-of-the-art background reference and to provide research support information for theoretical advancements and practical implementations of trusted computing on a mobile cloud. Mobile distributed cloud computing security and reliability issues are discussed. In particular, threat models are discussed as well as data/computation assignment approaches to a cloud composed of heterogeneous nodes. Presented solutions feature different levels of resiliency against malicious and misbehaving nodes. Further, benefits and pitfalls of different approaches are discussed over a set of different scenarios.

This chapter is organized as follows: the Introduction section has just described reliable distributed computing and it has motivated the remainder of this chapter. The Technology Background section discusses existing distributed computing approaches ranging from volunteer computing to clouds and mobile clouds. The State of the Art section reviews trusted computing approaches and tools over different platforms. It also highlights similarities and differences among them. The Trusted Distributed Mobile Computing section introduces the most relevant mobile trusted computing solutions, together with the hardware and software limitations of mobile platforms. It also discusses the main results of cheating-resilient computing approaches based on smart redundant computing. Further, it depicts differences among heterogeneous resource-constrained scenarios. The Practical hints and Future research directions section is aimed to help practitioners leverage latest technology and approaches to monitor and control remote execution. Further, this section highlights the most promising future trends for trusted distributed mobile computing. The Conclusion section summarizes the information contained in the previous sections and leads the reader to new perspectives of trusted autonomic computing approaches over a mobile cloud. Finally, the bibliography is reported, containing the most relevant works cited in this chapter, in order for the reader to be able to further delve into this exciting and relevant domain.

2 Technology Background

New smart devices are appearing every day, including wearable devices (glasses [43], watches [91]), automotive devices [89] or simply TV sets [45] and tablets [93]. Most if not all such devices have computing power that can be leveraged by a large set of novel applications (pay-per-distributed computing, healthcare, . . .).

Users are increasingly taking advantage of such novel hardware [77]. Such devices have not yet been used for reliable trustworthy computing, due to the lack of a generic reliability-guaranteeing approach and due to the lack of a mature implemented management framework. As a matter of fact, reliable distributed computing is known to be a challenging problem that continues to receive a special attention [28]. In fact, the pervasiveness of presently available networks and the availability of increasingly powerful computing devices offer new perspectives over the general problem depicted above.

As we already know, cloud computing offers unprecedented ways to pervasively access scalable and economically viable computing and storage resources. The encouraging results of volunteer computing projects in this context and the flexibility of the cloud, induces to spend research efforts towards new computing paradigms (one of the first examples was Cloud@Home [21]). In particular, the mobile cloud, defined as the possibility of offering services by also leveraging mobile nodes, has raised much interest in the research community [10, 60, 81]. A mobile cloud is the combination of cloud computing and mobile networks to enable new apps and services for mobile users, network operators, as well as cloud computing providers [1]. Mobile cloud computing aims at hosting complex applications on possibly a large number of mobile devices, leveraging heterogeneous computing resources and network technologies based on the pay-per-use principle. Mobile applications have a large number of open issues, concerned mainly with connectivity, computational capacities, memory and battery constraints.

Mobile multiplayer online games [34] are an example of fully distributed mobile approach. In mobile environments, several features might be exploited to enable resource sharing among multiple devices/game consoles owned by different mobile users. Trading computing/networking facilities among mobile players can be effective. This operation mode opens a wide number of interesting sharing scenarios. In particular, once mobile nodes make their resource available to the community, it becomes possible to distribute the workload for the management of the game itself [34].

Many successful attempts to efficiently distribute computation allow us to further discuss over the reliability and security of such approaches, with the aim to devise better solutions to the above problems. In particular, malicious software can alter affected cloud nodes' behavior and affect the reliability and security of the distributed effort. In the following subsection we analyze more in depth some characteristics of the attacks and the most relevant protection mechanisms.

2.1 Malware and Smart Devices

Thwarting malware attacks in smart devices is a challenging problem area with a substantial amount of open issues (summarized in Tables 1 and 2). Therefore, it is interesting to study and evaluate how such mobile distributed systems can autonomically [83] self-configure and adjust to attacks and to changes in the configuration [69]. In particular, as regards security, attacks targeting mobile devices are increasing in quantity and quality [6].

It is also worth noting that such devices often feature powerful sensing capabilities. In fact, they are capable of grabbing and monitoring sound, movement, position, heat, humidity, and presence via very sophisticated cameras. Further, one

Table 1 Smart devices opportunities and caveats

Devices	Opportunities	Caevats
Smartphones	Crowdsourcing [17]	Battery endurance, networking costs
Tablets	Crowdsourcing	Battery endurance, network availability
Wearables	Pervasive overall health monitoring	Malware and greyware
IMDs [25]	Pervasive health monitoring [37]	Targeted malware

Table 2 Smart devices and security issues

Devices	Security issues	Specific issues
Smartphones	Bloatware, malware and work-stealing-ware	Code and computation integrity
Tablets	Bloatware, malware and greyware	Code and computation integrity
IMDs [25]	Safety (e.g. pacemakers) and security	Alteration of vital parameters
Wearables	Mostly privacy issues	Personal data leakage

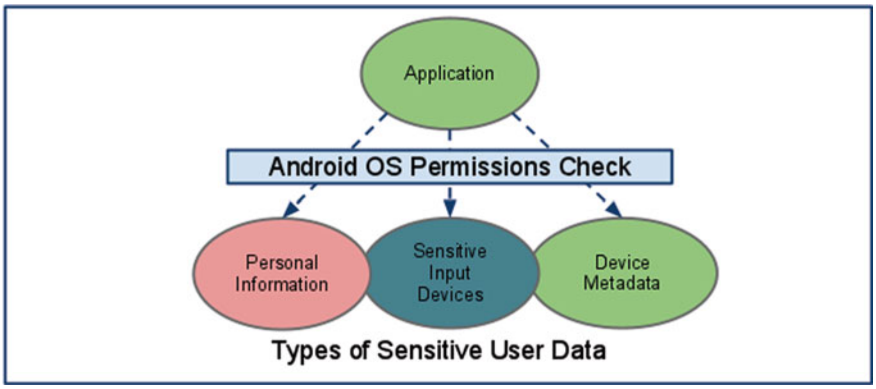


Fig. 1 High-level view of the Android permission system [40]

relevant feature of such devices is their ability to execute third-party apps from a large variety of markets [82]. This poses a number of security and privacy issues. In particular, software of malicious nature can access the services provided by the device and collect personal information and other private data. Further, it can also affect the behavior of other applications installed on the same device [77].

Present mobile malware detection techniques are similar to traditional desktop solutions. Unfortunately, they are not equally effective due to the limited computing resources and battery constraints of mobile devices. In practice, even though relevant improvements on malware detection for desktop and server PCs have been made, leveraging such techniques to mobile devices is still challenging.

The security architecture of present smartphones usually features a permission system restricting apps privileges [33, 44] (see also Fig. 1). However, this approach is not effective as apps request all permissions at install time, and the regular user is only left with the choice between accepting all of them altogether or denying access and not installing the app. In addition, apps can interact with each other and share information. Further, malware detection is quite difficult as signature-based antimalware techniques can only detect malware for which a signature is available, and are useless against polymorphic and metamorphic malware [76, 77].

Dynamic analysis based on behavior monitoring can be an effective approach for mobile nodes. One example is VirusMeter [55], a behavior-based malware detection approach based on the fact that mobile devices are usually battery powered and any malicious activity consumes precious battery power. VirusMeter relies on a power model characterizing power consumption of regular behavior and sending an alert when the power consumption pattern is anomalous. VirusMeter catches (mis)behaviors that lead to abnormal power consumption by monitoring power consumption of the mobile device.

3 State of the Art

The problem of secure reliable distributed computing has been modeled in a number of ways [47] that mostly focus on consistency of the global computation. This section in particular focuses on trustworthy outsourced computing, with some specific reference to mobile-hosted workloads. Relevant contributions come from a large number of heterogeneous areas as well as novel ideas and approaches that are discussed in the following taxonomy.

3.1 Verifiable Outsourced Computations

Two main objectives are relevant to verifiable outsourced computing: result correctness and timeliness of results. Gennaro et al. [36] are concerned with the former objective, whereas Moser et al. [61] with the latter. As discussed in [85], where proof-based verifiable computation is surveyed, the possibility of verifying computation while limiting or avoiding re-execution is quite a hot topic.

As a matter of fact, the efficient verification problem in distributed networks was investigated by Das Sarma [23] who proposed an effective distributed algorithm. Das Sarma introduced tight time lower bounds on distributed verification algorithms for many fundamental problems such as connectivity, spanning connected subgraph, and $s - t$ cut verification. The lower bound proofs he provided show interconnections between communication complexity and distributed computing. Sasson [13] gave another relevant contribution to the solution of the problem by discussing the efficiency of some Probabilistically-checkable Proofs. However, his results once again show that only a restricted set of problems can effectively use such approaches. Similarly, the work by Backes [9] addresses the reliable distributed computing problem for the class of computations of quadratic polynomials over a large number of variables. The downside of these approaches is that they are hardly general enough to be used for general-purpose computations. Some other more practical verification technique for outsourced computation was given in [71], leveraging Interactive Proofs [20].

An interesting approach is Blacksheep [14], a distributed system leveraging similarity amongst the machines for detecting the presence of malware among groups of similarly-configured machines. Golle [38] undoubtedly gave a relevant contribution towards addressing the reliable distributed computing problem by analyzing the motivations of cheating by untrusted computing resources. He proposed security schemes that protect against this threat by rendering cheating less convenient than well-behavior. In a way similar to Golle's, Belenkiy et al. [12] introduced a distributed checking mechanism where a reward is given to a contractor when he manages to detect and signal a cheating node. This is an interesting approach with a relevant issue. In fact, the contractors can themselves be malicious. As such, there is a chance of accusing/demoting noncheating nodes.

Wei et al. [88] proposed a privacy cheating discouragement and secure computation auditing protocol (SecCloud) which is a simple protocol bridging secure storage and secure computation auditing in cloud and achieving privacy cheating discouragement by designated verifier signature, batch verification and probabilistic sampling techniques. In particular, Wei et al. give an optimal sampling size to minimize the cost and suggest a practical secure-aware cloud computing testbed (SecHDFS).

A totally different solution was proposed by Di Martino [35], introducing an approach that can be used to perform complex event correlation analysis to identify intrusions that involve Service Level Agreement violations. Levitin et al. [52] present an algorithm for evaluating the reliability and distribution of complex non-repairable series-parallel multi-state systems with common cause failures caused by propagation of failures in system elements. Further, Levitin et al. [51] propose a general model for linear consecutive k-out-of-r-from-n system to the case of m consecutive runs of r elements. In this model, the system consists of n linearly ordered statistically independent elements, and fails *iff* in each of at least m consecutive overlapping groups of r consecutive elements at least k elements fail. This model can be useful in modeling failing or cheating computing nodes. Their approach can actually be useful to represent a malicious behavior of nodes that is caused by malware spreading among them.

It is worth noting that the main problem is usually due to node rational selfish behavior and not to malware. However, the above-described models can be useful in investigating/analyzing the effect of erroneous partial results on the final computed outcome.

As further examples of improved task distribution strategies, Su et al. [75] introduce an approach that optimizes the scheduling of tasks to nodes using a game-theoretic approach. ScaleStar [92] approaches many-task scheduling on clouds discussing a solution that especially takes into consideration resource cost. ScaleStar assigns the selected task to a virtual machine with higher comparative advantage which effectively balances the execution time-and-monetary cost goals. ScaleStar does not address the problem related to cheating nodes but it introduces novel interesting price-performance considerations on scheduling [19].

Further, Parno et al. introduce Pinocchio [65], i.e. a crypto-based system for efficient verification of remote computation. Pinocchio is an interesting specific approach using quadratic programs [41] for encoding computations and producing small-sized proof independently from the size of the computation. An interesting work similar to Pinocchio is [84] that considers interactive, proof-based verifiable computation and extends non-cryptographic protocols to a much broader class of computations. This work is relevant, as it the first one that takes into consideration Graphics Processing Units (GPU) as a useful computing resource for rapidly computing proofs.

A different but effective approach is taken by Cassadin et al. [18], considering preventive maintenance planning and production scheduling problems and proposing an integrated model that coordinates preventive maintenance planning decisions with single-machine scheduling decisions so that the total expected weighted completion time of jobs is minimized. Finally, Lopez et al. [58] discuss multiparty computation on the cloud and leverage a kind of homomorphic encryption to guarantee reliable execution.

3.2 Reputation Systems and Distributed Computing

A large number of reputation systems have been introduced in different fields to better rank and select items/people/restaurants/nodes. As for cloud nodes, Muralidharan [62] introduced a reputation system for volunteer cloud nodes based on performance [19], number of crashes and result correctness. Even though the basic idea is interesting, more advanced approaches can be found in terms of job replication strategy (just two-fold replication for them) and node selection. Furthermore, a more effective multi-ranking approach is needed to allow adjusting node selection to the user needs. As an example, *Cloud@Home* [21] is a simple but effective collaborative cloud node system.

Goodrich [39] studies pipelined algorithms for protecting distributed grid computations from cheating participants, who wish to be rewarded for tasks they receive but don't perform. Its cheater detection algorithms utilizes natural delays that exist in long-term grid computations. In particular, they partition the sequence of grid tasks into two interleaved sequences of task rounds, and use those rounds to devise a general-purpose scheme to detect cheaters. Goodrich and Eppstein [32] formalize the call combinatorial pair testing (CPT) problem, which has applications to the identification of uncooperative or unproductive participants in pair programming, massively distributed computing.

Yang investigates the problem of mapping computations to distributed remote nodes while guaranteeing the required SLA levels on Grids [67]. A similar approach can be adopted to efficiently use virtual machine resources [86] to guarantee adequate service levels. Du et al. [30] propose a commitment-based sampling scheme based on Merkle trees that proves more efficient than Golle's Ringer [38] when the function $f(x)$ to be computed is computationally costly, when it is nondeterministic or when the cardinality of the codomain of $f(x)$ is very small.

Finally, Differential Fault Analisis [76] is a dynamic analysis technique that aims to bridge the semantic gap and better classify code behavior by establishing cause-effect relationships between changes induced by the analyst and the visible effects they produce. This approaches offer advantages when the code to analyze is obfuscated, albeit it can have higher complexity and computing costs.

3.3 Autonomic Computing and Cloud

Autonomic Computing techniques can provide improved quality of service (QoS) management, power consumption management and overall better reliability and robustness of systems. As an example, a NAM Capacity Planner can be used [5] to obtain optimal energy consumption under different workloads, minimizing cost while maintaining functionality. The autonomic management of clouds has received a lot of attention by both academia and industry, with a special focus on IaaS platforms, while the self-management at the PaaS level becomes a major concern platforms as PaaS can be more difficult to manage and more rarely addressed.

Run-time monitoring and detection of critical situations is a fundamental requirement to achieve autonomic behavior in service-based cloud platforms. BioRAC [42] is a resilient autonomic cloud using multi-level tunable redundancy techniques to increase attack and exploitation resilience in cloud computing. This approach can also be useful against cheating nodes. A model for validating the convenience of cooperative cloud node strategies over selfish ones, where nodes do not help each other, is introduced in [4], describing the architecture of autonomous cloud platforms in a discrete-event simulator. Dautov et al. [24] present a vision of cloud application platforms as sensor networks, based on the similarities between the problem domain of cloud application platform monitoring and sensor-enabled domains as traffic surveillance, environmental monitoring or home automation.

A relevant autonomic system for cloud nodes that aims at creating a trusted cloud is *AntiCheetah* [27]. In *AntiCheetah* outsourcing costs are minimized and input elements are assigned to cloud nodes in an adaptive way according to evaluated past behavior of nodes. The approach is proven effective and resilient against simple misbehaving nodes (see a mobile use case in Fig. 2 [27]). The assignment of input elements to cloud nodes is performed in *AntiCheetah* as an autonomic, self-configuring system effective against various kinds of smart cheaters in the cloud. *AntiCheetah* leverages a multi-round approach to dynamically change the assignment matrix. *AntiCheetah* also detects and neutralizes smart cheaters that have complex cheating strategies. Its approach is effective and convenient under heterogeneous conditions. *AntiCheetah* is based on a previous approach, *CheR* [28] that was effectively used to model the assignment of work chunks to nodes as a Linear Programming problem. The two approaches above can effectively be combined in order to guarantee optimal distribution of workload at every iteration.

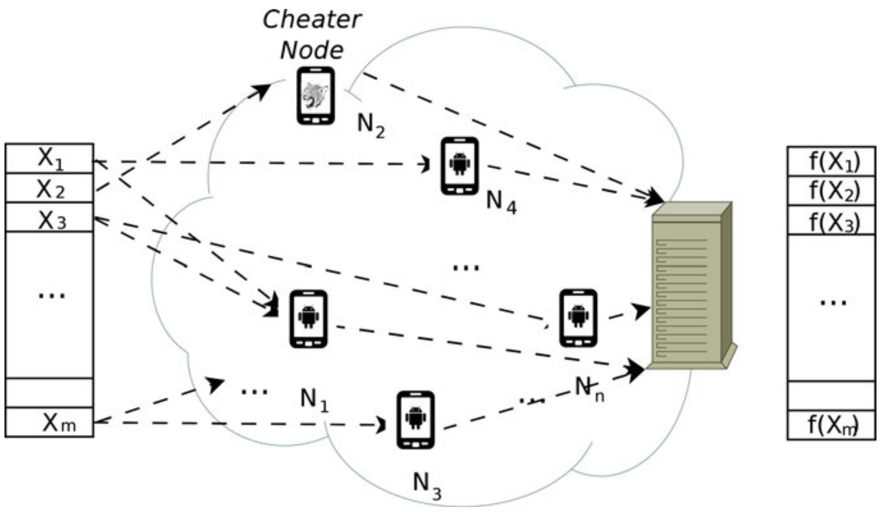


Fig. 2 A use case for a mobile cloud where computing tasks are distributed among possibly cheating mobile nodes

4 (Trusted) Distributed Mobile Computing

A large number of systems based on trusted computing techniques have been introduced so far for the cloud [72] and in mobile contexts [53, 90]. However, the trusted hardware platform on mobile devices has often been circumvented so far [63, 78]. The reason why is that efforts on mobile devices have mostly been focused on functionality and performance. However, only recently more powerful smart mobile devices have rendered the distributed mobile computing approach more feasible in practice.

Mobile distributed computing models have been proposed in the past [49], however interesting effective algorithms have only recently been introduced. As an example, Kim [48] investigated topology construction methods for heterogeneous distributed mobile computing with the aim to complete as many tasks as possible by their deadline using a distributed mobile computing system. As such, tasks have to be intelligently distributed among the devices to efficiently use the system resources. Kim proposed two topology algorithms to enhance the performance in comparison with the all-connected approach.

Mobile distributed computing has to take into consideration mobile node capabilities and topology [68]. In fact, autonomic wireless network configuration capabilities can help reducing transmission costs and improving reliability of communication and as such computation. Recently, this possibility of reconfiguration has been further enhanced with the introduction of Software Defined Networking (SDN) [3] and Network Function Virtualization (NFV) [70]. These approaches potentially allow improving effectiveness and efficiency of mobile clouds. However, capacity and latency issues can arise that can affect the convenience of the approach [11]. In addition, user-driven SDN can be vulnerable to misconfiguration and DoS [73]. As such these approaches have to be carefully investigated before deployment.

An example of an algorithm for efficient distributed mobile pattern matching has been introduced by Liu et al. [56], showing that effective distributed computing on a mobile cloud is possible. In this case data are not centrally stored. As such one person's pattern could be separately stored in a number of different nodes. This can potentially be vulnerable to fake computing attacks unless nodes are trusted. A trivial approach to pattern matching over a mobile environment would be to collect all the data distributed in base stations to a data center and conduct pattern matching at the data center afterwards. However, this approach can have a big impact on network traffic. Another smart distributed mobile algorithm has been proposed by Liu and Kang [56], showing an approach based on a weighted bloom filter (WBF [15]) that finds target patterns over a distributed mobile environment. In particular, Liu's approach offers successful pattern matching in distributed incomplete patterns, and it is provably resilient to cheating and misbehavior. In fact, only qualified IDs and corresponding weights in each base station are sent to the data center for aggregation and verification [56].

As with most distributed mobile computing approaches [54] the aim is towards reducing communication costs while replicating services over multiple coherent domains. A significant reduction in communication and computing power consumption is also targeted by Arnau [8]. Arnau leverages mobile GPUs in a power-efficient way by using Parallel Frame Rendering (PFR) techniques to effectively split the workload among battery-constrained devices. This novel approach is tailored towards a specific graphics-oriented application but can be effectively reused in a number of different GPGPU computing [59] scenarios [31, 50].

5 Practical Hints and Future Research Directions

Effective secure distributed computing on mobile nodes is still in its infancy. However, interesting solutions and approaches have already been contributed in the Literature [27, 42, 46]. At present, the most sensible approach consists in adopting and adapting state-of-the-art solutions for both mobile node security and integrity [76] as well as novel algorithms aimed at guaranteeing the computation is not affected by malware or misbehavior [27]. The practitioner willing to leverage such opportunity can migrate and deploy some of the existing approaches on a Java-supporting mobile device cloud [66], with an eye to power efficiency and fair balancing of workloads.

As mobile devices get more powerful and efficient [29] it becomes of vital importance to expose such resources in the most effective, efficient and secure way. In particular, mobile GPU computing [8, 87] (see Fig. 3) is a rising trend that will contribute much to the future of mobile cloud computing. A number of competing approaches have been introduced: Apple's Mobile OpenCL [16],

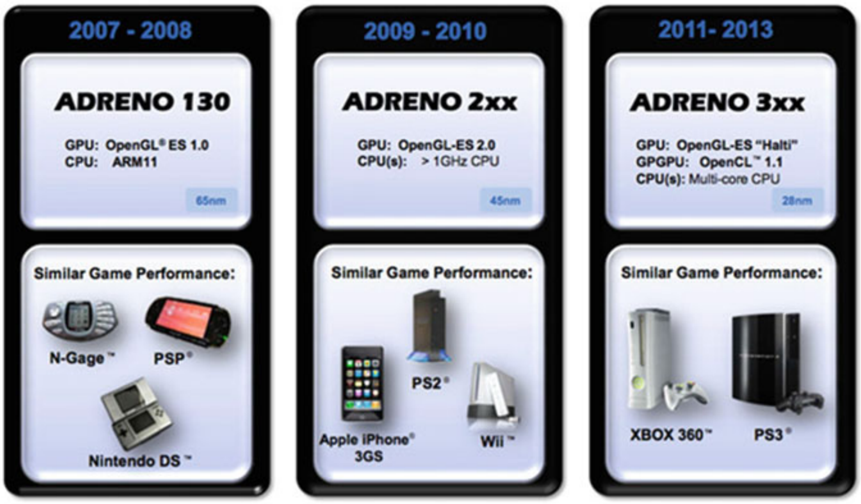


Fig. 3 An overview of mobile GPU evolution related to OpenCL language support [74]

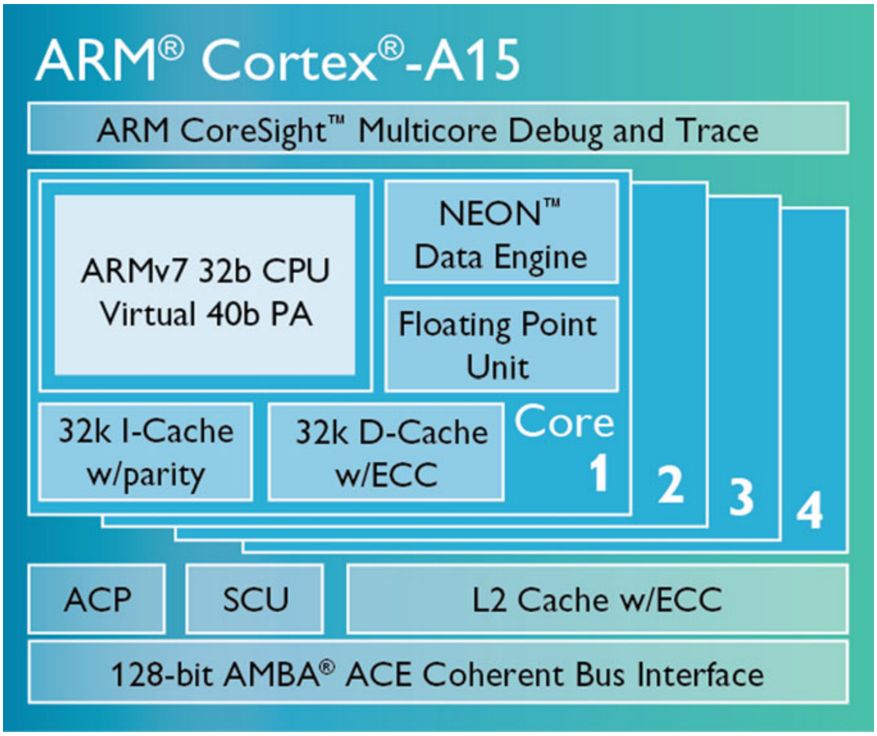


Fig. 4 Architecture of an hardware virtualization supporting mobile CPU [7]

Google's Renderscript [26] and NVIDIA's CUDA [64]. The final winner of this contest will shape the future mobile-GPU-as-a-service offerings. This is closely tied to the support GPU vendors and mobile OS implementors will be able to give to mobile distributed GPGPU computing. Naturally, GPUs will also be increasingly useful in checking the integrity of CPU code and data [57], in order to better protect the mobile node itself. This way, code and data integrity can be offered in a power-efficient way, leveraging parallel processing power of manycores [79].

In addition, following Personal Computing evolution, virtualization and sand-boxing/monitoring for security will be increasingly popular on the new 64-bit hardware-virtualization-supporting ARM CPUs [7, 22], as depicted in Fig. 4 where the components of new Cortex A15 platform are visible. Mobile virtualization [80] will be increasingly common and it will help better isolating both local and remote distributed workloads. In conjunction with GPGPU parallel processing power, this will further foster the development of novel advanced applications in the mobile application markets.

6 Conclusion

Even though reliable distributed computing has long been a hot research topic, effective and efficient mobile distributed computing has not been achieved yet. However, the increased processing power and networking/storage capacity of mobile computing devices has motivated their inclusion in cloud computing systems. A very fast spreading of novel applications can be foreseen, enabling secure multiparty computations over heterogeneous mobile nodes. Further enhanced mechanism for distributing the workload in such an environment show that mobile cloud computing is a worthwhile approach but it has much room for improvement. In particular, the autonomic reliable execution of parallel workload over a large number of mobile heterogeneous cheating nodes is a real challenge, especially when malicious and/or cheating nodes and smart cheating strategies are in place.

Novel approaches that integrate and improve over previous reliable distributed computing solutions are needed. They will have to leverage novel technologies such as mobile GPU computing. Future research will need to devise smarter approaches to address the above-defined problems. This will allow to make effective use of the large pervasively available, cheap and powerful computing resources without affecting mobile device functionality and resource consumption.

References

1. Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., Buyya, R.: Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges. IEEE Commun. Surv. Tutorials **16**(1), 337–368 (2014)

2. Agarwal, A., Govindaraj, J., Juneja, N., Naik, V.: Feasibility study of on-device and in-the-cloud virtualization of mobiles. In: Proceedings of the 5th IBM Collaborative Academia Research Exchange Workshop, I-CARE '13, pp. 5:1–5:4. ACM, New York (2013)
3. Akyildiz, I.F., Lee, A., Wang, P., Luo, M., Chou, W.: A roadmap for traffic engineering in sdn-openflow networks. Comput. Netw. **71**, 1–30 (2014)
4. Amoretti, M., Lafuente, A.L., Sebastio, S.: A cooperative approach for distributed task execution in autonomic clouds. In: 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008), pp. 0:274–281 (2013)
5. Amoretti, M., Zanichelli, F., Conte, G.: Efficient autonomic cloud computing using online discrete event simulation. J. Parallel Distrib. Comput. **73**(6), 767–776 (2013)
6. Arabo, A., Pranggono, B.: Mobile malware and smart device security: Trends, challenges and solutions. In: Proceedings of the 2013 19th International Conference on Control Systems and Computer Science, CSCS '13, pp. 526–531. IEEE Computer Society, Washington (2013)
7. ARM: Cortex-a15 processor (2014). http://www.arm.com/products/processors/cortex-a/cortex-a15.php
8. Arnau, J.-M., Parcerisa, J.-M., Xekalakis, P.: Parallel frame rendering: Trading responsiveness for energy on a mobile gpu. In: Proceedings of the 22Nd International Conference on Parallel Architectures and Compilation Techniques, PACT '13, pp. 83–92. Piscataway (2013)
9. Backes, M., Fiore, D., Reischuk, R.M.: Verifiable delegation of computation on outsourced data. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security, CCS '13, pp. 863–874. ACM, New York (2013)
10. Bahl, P., Han, R.Y., Li, L.E., Satyanarayanan, M.: Advancing the state of mobile cloud computing. In: Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services, MCS '12, pp. 21–28. ACM, New York (2012)
11. Basta, A., Kellerer, W., Hoffmann, M., Morper, H.J., Hoffmann, K.: Applying nfv and sdn to lte mobile core gateways, the functions placement problem. In: Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, &#38; Challenges, AllThingsCellular '14, pp. 33–38. ACM, New York (2014)
12. Belenkiy, M., Chase, M., Erway, C.C., Jannotti, J., Küpçü, A., Lysyanskaya, A.: Incentivizing outsourced computation. In: Proceedings of the 3rd International Workshop on Economics of Networked Systems, NetEcon '08, pp. 85–90. ACM, New York (2008)
13. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E.: On the concrete efficiency of probabilistically-checkable proofs. In: Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing, STOC '13, pp. 585–594. ACM, New York (2013)
14. Bianchi, A., Shoshitaishvili, Y., Kruegel, C., Vigna, G.: Blacksheep: Detecting compromised hosts in homogeneous crowds. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12, pp. 341–352. ACM, New York (2012)
15. Bruck, J., Gao, J., Jiang, A.: Weighted bloom filter. In: 2006 IEEE International Symposium on Information Theory pp. 2304–2308 (2006)
16. Bucur, A.: Opencl - opengl es interop: Processing live video streams on a mobile device - case study. In: ACM SIGGRAPH 2013 Mobile, SIGGRAPH '13, pp. 15:1–15:1. ACM, New York (2013)
17. Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: behavior-based malware detection system for Android. In: Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM '11, pp. 15–26. ACM, New York (2011)
18. Cassady, C., Kutanoglu, E.: Integrating preventive maintenance planning and production scheduling for a single machine. IEEE Trans. Reliab. **54**(2), 304–309 (2005)
19. Chidambaram Nachiappan, N., Yedlapalli, P., Soundararajan, N., Kandemir, M.T., Sivasubramaniam, A., Das, C.R.: Gemdroid: A framework to evaluate mobile platforms. SIGMETRICS Perform. Eval. Rev. **42**(1), 355–366, (2014
20. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12, pp. 90–112. ACM, New York (2012)

21. Cunsolo, V., Distefano, S., Puliafito, A., Scarpa, M.: Volunteer computing and desktop cloud: The cloud@home paradigm. In: Eighth IEEE International Symposium on Network Computing and Applications, 2009. NCA 2009, pp. 134–139 (2009)
22. Dall, C., Nieh, J.: Kvm/arm: The design and implementation of the linux arm hypervisor. In: Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '14, pp. 333–348. ACM, New York (2014)
23. Das Sarma, A., Holzer, S., Kor, L., Korman, A., Nanongkai, D., Pandurangan, G., Peleg, D., Wattenhofer, R.: Distributed verification and hardness of distributed approximation. In: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, STOC '11, pp. 363–372. ACM, New York (2011)
24. Dautov, R., Paraskakis,I.: A vision for monitoring cloud application platforms as sensor networks. In: Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference, CAC '13, pp. 25:1–25:8. ACM, New York (2013)
25. Denning, T., Borning, A., Friedman, B., Gill, B.T., Kohno, T., Maisel, W.H.: Patients, pacemakers, and implantable defibrillators: Human values and security for wireless implantable medical devices. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '10, pp. 917–926, ACM, New York (2010)
26. Developers, A.: Renderscript. http://developer.android.com/guide/topics/renderscript/compute.html.
27. Di Pietro, R., Lombardi, F., Martinelli, F., Sgandurra, D.: Anticheetah: An autonomic multi-round approach for reliable computing. In: Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC), pp. 371–379 (2013). (Best Paper Award)
28. Di Pietro, R., Lombardi, F., Martinelli, F., Sgandurra, D.: CheR: Cheating Resilience in the Cloud via Smart Resource Allocation. In: Danger, J.L., Debbabi, M., Marion, J.-Y., Garcia-Alfaro, J., Zincir Heywood, N. (eds.), Foundations and Practice of Security, Lecture Notes in Computer Science, pp. 339–352. Springer International Publishing, Switzerland (2014)
29. Djatmiko, M., Cunche, M., Boreli, R., Seneviratne, A.: Heterogeneous secure multi-party computation. In: Proceedings of the 11th International IFIP TC 6 Conference on Networking - Volume Part II, IFIP'12, pp. 198–210. Springer, Berlin/Heidelberg (2012)
30. Du, W., Murugesan, M., Jia, J.: Algorithms and Theory of Computation Handbook, Chapter Uncheatable Grid Computing, pp. 30–30. Chapman and Hall/CRC, London (2010)
31. Duarte, S., Navalho, D., Ferreira, H., Preguiça, N.: Scalable data processing for community sensing applications. Mob. Netw. Appl. **18**(3), 357–372 (2013)
32. Eppstein, D., Goodrich, M.T., Hirschberg, D.S.: Combinatorial pair testing: distinguishing workers from slackers. In: Proceedings of the 13th International Conference on Algorithms and Data Structures, WADS'13, pp. 316–327, Springer, Berlin/Heidelberg (2013)
33. Felt, A.P., Wang, H.J., Moshchuk, A., Hanna, S., Chin, E.: Permission re-delegation: Attacks and defenses. In: Proceedings of the 20th USENIX Conference on Security, SEC'11, pp. 22–22. USENIX Association, Berkeley (2011)
34. Ferretti, S., D'Angelo, G.: Mobile online gaming via resource sharing. In: Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques, SIMUTOOLS '12, pp. 262–269, ICST, Brussels, Belgium (2012) (ICST Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
35. Ficco, M., Rak, M., Di Martino, B.: An intrusion detection framework for supporting sla assessment in cloud computing. In: 2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN), pp. 244–249 (2012)
36. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: Proceedings of the 30th Annual Conference on Advances in Cryptology, CRYPTO'10, pp. 465–482. Springer, New York Berlin/Heidelberg (2010)
37. Gisdakis, S., Giannetsos, T., Papadimitratos, P.: Sppear: Security &#38; privacy-preserving architecture for participatory-sensing applications. In: Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless &#38; Mobile Networks, WiSec '14, pp. 39–50. ACM, New York (2014)

38. Golle, P., Mironov, I.: Uncheatable distributed computations. In: Proceedings of the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA, CT-RSA 2001, pp. 425–440. Springer, New York (2001)
39. Goodrich, M.T.: Pipelined algorithms to detect cheating in long-term grid computations. Theor. Comput. Sci. **408**(2-3), 199–207 (2008)
40. Google: Android security overview (2014). http://source.android.com/devices/tech/security
41. Groenwold, A.A.: Positive definite separable quadratic programs for non-convex problems. Struct. Multidiscip. Optim. **46**(6), 795–802 (2012)
42. Hariri, S., Eltoweissy, M., Al-Nashif, Y.: Biorac: biologically inspired resilient autonomic cloud. In: Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research, CSIIRW '11, pp. 80:1–80:1. ACM, New York (2011)
43. Hong, J.: Considering privacy issues in the context of google glass. Commun. ACM **56**(11), 10–11 (2013)
44. Jeon, J., Micinski, K.K., Vaughan, J.A., Fogel, A., Reddy, N., Foster, J.S., Millstein, T.: Dr. android and mr. hide: Fine-grained permissions in android applications. In: Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM '12, pp. 3–14. ACM, New York (2012)
45. Joo, Y., Lee, D., Kim, J., Eom, Y.I.: Cgroups-based scheduling scheme for heterogeneous workloads in smart tv systems. In: Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, ICUIMC '13, pp. 96:1–96:5. ACM, New York (2013)
46. Kakadia, D., Saripalli, P., Varma, V.: Mecca: Mobile, efficient cloud computing workload adoption framework using scheduler customization and workload migration decisions. In: Proceedings of the First International Workshop on Mobile Cloud Computing &#38; Networking, MobileCloud '13, pp. 41–46. ACM, New York (2013)
47. Kalyon, G., Le Gall, T., Marchand, H., Massart, T.: Symbolic supervisory control of distributed systems with communications. IEEE Trans. Autom. Control **59**(2), 396–408 (2014)
48. Kim, I.-Y., Kim, J.-K.: Enhancing the performance of a distributed mobile computing environment by topology construction. In: Proceedings of the 12th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part II, ICA3PP'12, pp. 21–30. Springer, Heidelberg/Berlin (2012)
49. Kurkovsky, S., Bhagyavati, M.S., Ray, A.: A collaborative problem-solving framework for mobile devices. In: Proceedings of the 42Nd Annual Southeast Regional Conference, ACM-SE 42, pp. 5–10. ACM, New York (2004)
50. Lee, K., Lee, J., Yi, Y., Rhee, I., Chong, S.: Mobile data offloading: How much can wifi deliver? IEEE/ACM Trans. Netw. **21**(2), 536–550 (2013)
51. Levitin, G., Dai, Y.: Linear m -consecutive k -out-of- r -from- n:f systems. IEEE Trans. Reliab. **60**(3), 640–646 (2011)
52. Levitin, G., Xing, L., Ben-Haim, H., Dai, Y.: Reliability of series-parallel systems with random failure propagation time. IEEE Trans. Reliab. **62**(3), 637–647 (2013)
53. Liang, W.-Y., Hsieh, Y.-M., Lyu, Z.-Y.: Design of a dynamic distributed mobile computing environment. In: 2007 International Conference on Parallel and Distributed Systems, vol. 2, pp. 1–8 (2007)
54. Lin, F.X., Wang, Z., Zhong, L.: K2: A mobile operating system for heterogeneous coherence domains. SIGARCH Comput. Archit. News **42**(1), 285–300 (2014)
55. Liu, L., Yan, G., Zhang, X., Chen, S.: Virusmeter: Preventing your cellphone from spies. In: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, RAID '09, pp. 244–264. Springer, New York, Berlin/Heidelberg (2009)
56. Liu, S., Kang, L., Chen, L., Ni, L.: How to conduct distributed incomplete pattern matching. IEEE Trans. Parallel Distrib. Syst. **25**(4), 982–992 (2014)
57. Lombardi, F., Di Pietro, R.: CUDACS: securing the cloud with CUDA-enabled secure virtualization. In: Proceedings of the 12th international conference on Information and communications security, ICICS'10, pp. 92–106. Springer, Berlin/Heidelberg (2010)

58. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the 44th Symposium on Theory of Computing, STOC '12, pp. 1219–1234. ACM, New York (2012)
59. Luebke, D., Harris, M., Krüger, J., Purcell, T., Govindaraju, N., Buck, I., Woolley, C., Lefohn, A.: Gpgpu: general purpose computation on graphics hardware. In: SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes, pp. 33. ACM, New York (2004)
60. Luo, L., Wu, W., Di, D., Zhang, F., Yan, Y., Mao, Y.: A resource scheduling algorithm of cloud computing based on energy efficient optimization methods. In: Proceedings of the 2012 International Green Computing Conference (IGCC), IGCC '12, pp. 1–6. IEEE Computer Society, Washington (2012)
61. Moser, H.: Towards a real-time distributed computing model. Theor. Comput. Sci. **410**(6-7), 629–659 (2009)
62. Muralidharan, S., Kumar, V.: A novel reputation management system for volunteer clouds. In: 2012 International Conference on Computer Communication and Informatics (ICCCI), pp. 1–5 (2012)
63. Nadkarni, A., Tendulkar, V., Enck, W.: Nativewrap: Ad hoc smartphone application creation for end users. In: Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks, WiSec '14, pp. 13–24. ACM, New York (2014)
64. NVIDIA: Cuda for arm platforms is now available. http://devblogs.nvidia.com/parallelforall/cuda-arm-platforms-now-available.
65. Parno, B., Gentry, C., Howell, J., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: Procedings of the 34th IEEE Symposium on Security and Privacy (2013)
66. Picco, G.P., Julien, C., Murphy, A.L., Musolesi, M., Roman, G.-C.: Software engineering for mobility: Reflecting on the past, peering into the future. In: Proceedings of the on Future of Software Engineering, FOSE 2014, pp. 13–28. ACM, New York (2014)
67. Quan, D.M., Yang, L.T.: Parallel mapping with time optimization for sla-aware compositional services in the business grid. IEEE Trans. Serv. Comput. **4**(3), 196–206 (2011)
68. Ryoo, J., Kim, H.: Multi-sector multi-range control for self-organizing wireless networks. J. Netw. Comput. Appl. **34**(6), 1848–1860 (2011)
69. Samimi, F.A., McKinley, P.K., Sadjadi, S.M.: Mobile service clouds: A self-managing infrastructure for autonomic mobile computing services. In: Proceedings of the Second IEEE International Conference on Self-Managed Networks, Systems, and Services, SelfMan'06, pp. 130–141. Springer, Berlin, Heidelberg (2006)
70. Sapio, A., Liao, Y., Baldi, M., Ranjan, G., Risso, F., Tongaonkar, A., Torres, R., Nucci, A.: Per-user policy enforcement on mobile apps through network functions virtualization. In: Proceedings of the 9th ACM Workshop on Mobility in the Evolving Internet Architecture, MobiArch '14, pp. 37–42. ACM, New York (2014)
71. Setty, S., Blumberg, A.J., Walfish, M.: Toward practical and unconditional verification of remote computations. In: Proceedings of HotOS XIII. Usenix (2011)
72. Shen, Z., Li, L., Yan, F., Wu, X.: Cloud computing system based on trusted computing platform. In: 2010 International Conference on Intelligent Computation Technology and Automation (ICICTA), vol. 1, pp. 942–945 (2010)
73. Shin, S., Yegneswaran, V., Porras, P., Gu, G.: Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security, CCS '13, pp. 413–424. ACM, New York (2013)
74. SoC: Qualcomm chipsets vs performance. http://www.insidehardware.it/mobile/smart-phone/2911-htc-one-alternativa-convincente?start=4#.U8zgl3V53UY (2011)
75. Su, S., Li, J., Huang, Q., Huang, X., Shuang, K., Wang, J.: Cost-efficient task scheduling for executing large programs in the cloud. Parallel Comput. **39**(4–5), 177–188 (2013)
76. Suarez-Tangil, G., Tapiador, J.E., Lombardi, F., Pietro, R.D.: Thwarting obfuscated malware via differential fault analysis. Computer **47**(6) 24–31 (2014)
77. Suarez-Tangil, G., Tapiador, J.E., Peris, P., Ribagorda, A.: Evolution, detection and analysis of malware for smart devices. IEEE Commun. Surv. Tutorials **99**, 1–27 (2013)

78. Sun, M., Tan, G.: Nativeguard: Protecting android applications from third-party native libraries. In: Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless &#38; Mobile Networks, WiSec '14, pp. 165–176. ACM, New York (2014)
79. Tilli, A., Bartolini, A., Cacciari, M., Benini, L.: Don't burn your mobile!: Safe computational re-sprinting via model predictive control. In: Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS '12, pp. 373–382. ACM, New York (2012)
80. Varanasi, P., Heiser, G.: Hardware-supported virtualization on arm. In: Proceedings of the Second Asia-Pacific Workshop on Systems, APSys '11, pp. 11:1–11:5. ACM, New York (2011)
81. Vemulapalli, C., Madria, S.K., Linderman, M.: Pre-distribution scheme for data sharing in mobile cloud computing. In: Proceedings of the First International Workshop on Mobile Cloud Computing &#38; Networking, MobileCloud '13, pp. 11–18. ACM, New York (2013)
82. Vidas, T., Christin, N.: Sweetening android lemon markets: Measuring and combating malware in application marketplaces. In: Proceedings of the Third ACM Conference on Data and Application Security and Privacy, CODASPY '13, pp. 197–208. ACM, New York (2013)
83. Viswanathan, H., Lee, E.K., Rodero, I., Pompili, D.: An autonomic resource provisioning framework for mobile computing grids. In: Proceedings of the 9th International Conference on Autonomic Computing, ICAC '12, pp. 79–84. ACM, New York (2012)
84. Vu, V., Setty, S., Blumberg, A., Walfish, M.: A hybrid architecture for interactive verifiable computation. In: 2013 IEEE Symposium on Security and Privacy (SP), pp. 223–237 (2013)
85. Walfish, M.: Verifying the correctness of remote executions: From wild implausibility to near practicality. In: Proceedings of the 9th Workshop on Hot Topics in Dependable Systems, HotDep '13, pp. 7:1–7:1. ACM, New York (2013)
86. Wan, J., Yang, L.T., Li, Y., Xu, X., Xiong, N.: An adaptive management mechanism for resource scheduling in multiple virtual machine system. In: Calero, J., Yang, L., Màrmol, F., Garcìa Villalba, L., Li, A., Wang, Y. (eds.) Autonomic and Trusted Computing, vol. 6906 of Lecture Notes in Computer Science, pp. 60–74. Springer, Berlin/Heidelberg (2011)
87. Wang, Y.-C., Donyanavard, B., Cheng, K.-T.T.: Energy-aware real-time face recognition system on mobile cpu-gpu platform. In: Proceedings of the 11th European Conference on Trends and Topics in Computer Vision - Volume Part II, ECCV'10, pp. 411–422. Springer, Berlin/Heidelberg (2012)
88. Wei, L., Zhu, H., Cao, Z., Dong, X., Jia, W., Chen, Y., Vasilakos, A.V.: Security and privacy for storage and computation in cloud computing. Inform. Sci. **258**, 371–386 (2014)
89. Wilfinger, D., Murer, M., Baumgartner, A., Döttlinger, C., Meschtscherjakov, A., Tscheligi, M.: The car data toolkit: Smartphone supported automotive hci research. In: Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications, AutomotiveUI '13, pp. 168–175. ACM, New York (2013)
90. Wu, X., Zhang, H., Shen, Z.: Integrity measurement enhanced security for mobile agent based on trusted computing platform. In: WiCOM '08. 4th International Conference on Wireless Communications, Networking and Mobile Computing, 2008, pp. 1–4 (2008)
91. Xu, Y., Stojanovic, N., Stojanovic, L., Kostic, D.: An approach for dynamic personal monitoring based on mobile complex event processing. In: Proceedings of International Conference on Advances in Mobile Computing &#38; Multimedia, MoMM '13, pp. 464:464–464:473. ACM, New York (2013)
92. Zeng, L., Veeravalli, B., Li, X.: Scalestar: Budget conscious scheduling precedence-constrained many-task workflow applications in cloud. In: Proceedings of the 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, AINA '12, pp. 534–541. IEEE Computer Society, Washington (2012)
93. Zhao, Z., Hwang, K., Villeta, J.: Game cloud design with virtualized cpu/gpu servers and initial performance results. In: Proceedings of the 3rd Workshop on Scientific Cloud Computing Date, ScienceCloud '12, pp. 23–30. ACM, New York (2012)

Infiltrating Social Network Accounts: Attacks and Defenses

Rahul Potharaju, Bogdan Carbunar, Mozhgan Azimpourkivi, Venugopal Vasudevan, and S.S. Iyengar

Abstract Social networks collect and make public an unprecedented amount of user location information. This raises significant user privacy concerns. In this chapter, we describe social networking infiltration attacks, where adversaries befriend random victims of their choice and acquire access to their private information. To address this problem, we propose verification mechanisms that use location information certified by geosocial networks to quantify the context shared by a user and an inviter. We develop novel visual notifications that leverage the outcome of the proposed verifications to inform users about the context they share with inviters. The impact of location information makes it however vulnerable to attacks: users can rely on existing tools to report fraudulent locations. We describe venue centric location verification solutions that are resilient to powerful adversaries.

1 Introduction

Online Social Networks (OSNs) such as Facebook have become ubiquitous in the past few years, counting hundreds of millions of people as members. OSNs allow users to form friendship relationships, join groups, communicate and share information with friends. Most social network users are likely to be well behaved. However, the amount and ease of accessibility of personal information (e.g., date of birth, location, status updates) available on such sites is likely to draw a wide range of users with a malicious intent. Information collected from unsuspecting users can be used to subsequently launch targeted attacks such as spear phishing [1] and spamming [2] attacks.

R. Potharaju
Purdue University, West Lafayette, IN 47907, USA

Microsoft, Redmond, WA 98074, USA

B. Carbunar (✉) • M. Azimpourkivi • S.S. Iyengar
Florida International University, Miami, FL 33199, USA
e-mail: carbunar@cs.fiu.edu; mozhganaz@gmail.com; iyengar@cs.fiu.edu

V. Vasudevan
Arris, Palatine, IL 60038, USA

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_15

In this chapter we focus on *infiltration* attacks, launched by malicious users who send friend invitations to specific targets. When a target accepts the invitation, her personal account information is implicitly shared with the inviter. Befriending users further provides a permanent channel into their existence. The attackers can monitor the activities of their friends, influence their behavior, learn detailed preferences (e.g., through polls, group join invitations, etc) and even send them information (e.g., ads, fake reviews, news and spam).

Most of these attempts have financial gain as their ultimate goal. Spammers send out mass advertisements to a large number of users in hopes of selling their products. Phishers, on the other hand, attempt to fraudulently acquire sensitive information from a victim by impersonating a trusted third party (e.g. a banking corporation). In a study by Gartner [3], about 19 % of all those surveyed reported having clicked on a link in a phishing email, and 3 % admitted to giving up financial or personal information. This reasonable yield, despite having little information about the target victim, suggests that the effect can be more serious when additional victim information is available.

In this chapter, we show that attackers can obtain social networking information by infiltrating OSNs using Sybils, fake accounts they control. We identify several attacks and formalize a novel *3-Clique* Attack, to establish and leverage common context with victims in order to infiltrate even tightly knit communities.

We propose an attack system called iFriendU, a back-end driver written in Java to control Mozilla Firefox through Javascript code injection. We use iFriendU to demonstrate the seriousness of the 3-Clique attack on more than 10,000 Facebook users over a period of 6 months. Our experiments show that the 3-Clique attack can be easily automated, easy to perform and efficient—up to 78 % of targets fall victims, exceeding by 75 % the effectiveness of existing attacks. Moreover, we show that persistence pays off—repeating the same attack may turn rejects into accepts and that sharing more friends with the victim increases the attacker's success rate.

Efficient infiltration detection techniques cannot rely on the appearance of user accounts. Attacker accounts can be fake, filled with seemingly meaningful information. Pictures, opinions and friends are easy to fabricate and social botnets [4] can be used to make them seem active.

Instead, we propose an infiltration detection solution based on location information certified by geosocial networks. We use past location information to validate location claims made by users in their profiles, and evaluate the number of past co-location events between two users. Real-time verifications rely on instantaneous location information, i.e., signaling current co-location events. These verifications are intended to gauge the correctness of location claims made by an inviter in her public profile, determine location affinity levels between an inviter and an invitee (the invited user) and to elicit user feedback based on detected co-location events with the inviter. Furthermore, we develop novel visual notifications that leverage the outcome of the verifications to inform users about the context they share with inviters.

The solution proposed relies on the correctness of location information claimed by users. This is a concern, as malicious social network users have been shown to falsify their location trajectory, by claiming locations where they have not been present.[1] Rewards provided by participating franchises (e.g., Ann Taylor, GAP, Lufthansa, Starbucks, Pizza Hut) to users that frequently check-in at their locations, provide financial incentives for committing *location fraud*. The recent emergence of specialized GPS cheating applications for the most popular mobile eco-systems (e.g., LocationSpoofer [6] for iPhone and GPSCheat [7] for Android) simplifies location fraud: He et al. [8] proved the simplicity of performing fake check-ins in Foursquare.

To prevent location fraud, we propose to require users to present proofs of locations where they checked-in. To build such proofs, we introduce a suite of venue-oriented, secure location verification mechanisms. Our verifications require participating venues to deploy small, portable devices. To promote its adoptability, we design the verifications to be not only secure and correct, but also user friendly, low-cost and easy to deploy. We propose mechanisms that display QR codes encoding venue certified information, and that implement challenge/response protocols.

The remaining of the chapter is organized as follows.

2 Background and Model

2.1 Overview of Social Networks

We model a social network as an undirected graph $G = (V, E)$, where the nodes V represent the registered users and the edges E represent friend relations. We use $e = (u, v)$ to denote the existence of a friendship relation $e \in E$ between two users $u, v \in V$. Without loss of generality, we take the particular case of Facebook but it should be noted that our discussion applies to other OSNs as well. We emphasize the following properties of the social network:

Ease of Registration Social networks store an *account* for each registered user. We assume that it is easy for anyone to register a user account. Registering in today's OSNs requires the user to solve a challenge-response CAPTCHA [9] (Completely Automated Public Turing test to tell Computers and Humans Apart). Even though protection schemes like (re)CAPTCHAs and confirmation e-mails may be included, the human costs (money and time) for registering are insignificant. Note however that such protection schemes may make it more difficult but not impossible to massively register accounts.

Friendship and Messaging Users can establish friendship relations by extending an *invitation*. Once an invitation is sent from a user v to a user u, u may accept ("*Confirm*" in Facebook), reject ("*Ignore*" in Facebook) or leave the invitation

[1]Foursquare CEO admits 2–3 % of check-ins are fake [5].

pending by not giving a response. If and only if u decides to accept the invitation, is u's profile shared with v. Users also have the ability to send messages to other users, even to non-friends. However, similar to invitations, when a user u sends a message to a user v, u's profile is revealed to v.

Account Data As mentioned above, each user has an account, that includes personal profile information, such as name, date of birth, picture, e-mail and snail mail address but also other items such as wall postings, tagged pictures and videos.

Default Access Control Settings When a user registers an account it can choose the access permissions to each of its profile's components—who can access which fields of the user's account. While users have the option of changing the default settings, we have noticed in our experiments that many users have not used it.

2.2 Overview of Geosocial Networks

Most GSNs provide similar functionality: Users *check-in* at *venues* where they are present, effectively reporting their location to the geosocial network provider. As a reward, users receive *badges* and *mastership's* (or *virtual items* in Gowalla) as well as financial rewards. Franchises like Ann Taylor, GAP, Lufthansa, Starbucks and Pizza Hut have modified their business model to offer substantial discounts to users performing frequent check-ins. The functionality of geosocial networks centers on the following functionality.

Venues and Check-Ins The provider supports a given set of locations, defined in terms of discrete points-of-interests (POIs) or sites: restaurants, dentist offices, etc. During a check-in, the user's application (client) captures the GPS location and displays a list of close-by venues—the user can choose one. In the following, we use the term *check-in venue* to refer to a venue where a check-in is claimed to be performed. We call a *fake check-in* to be a check-in performed when the user is not physically located at the check-in venue.

Location Verifications An excellent example of security by obscurity, location verification mechanisms are kept secret by GSN providers. However, once attackers discover the nature and parameters of these verifications, they can easily circumvent them (e.g., see He et al. [8]).

2.3 System Model

We consider a system that consists of a social network provider, S. S hosts the system and serves a number of subscribers. We also consider S to be a *geosocial network*, e.g., Foursquare [10], Yelp [11], that extends the social network with location based experiences.

We assume users have mobile devices equipped with a GPS receiver and a Wi-Fi interface (present on most smartphones). To use several provider services, a client application needs to be downloaded and installed. Registered users receive initial service credentials, including a unique user id; let Id_A denote the id of user A.

The users rely on this app to report their location, through *check-ins* at venues of interest, share it with friends and be awarded "points" and "badges" (e.g., "Adventurer", "Explorer", or "Superstar"). A user earns a badge when it accumulates a certain number of check-ins, at the same or different venues.

In the following, the user that sends an invitation is said to be the *inviter* and the user receiving the invitation is the *invitee*. A *check-in venue* is a venue where a user has performed a CheckIn operation. We use the notation $F(A(param_A), B(param_B))$ to denote a procedure F executed between two participants A and B, each with its own input parameters.

2.4 Attacker Model

We assume that the social network provider is semi-honest (honest but curious): it will run its part of the protocols correctly, but it will try to learn private information from its users. We assume that users can be malicious. In the following, we describe several malicious behaviors that can be exhibited by attackers.

Sybil Account Creation Malicious users are assumed to be able to register and control an arbitrary number of fake, Sybil user accounts.

Information Seekers Such malicious users seek to invade the privacy of other users. We broadly classify such malicious users into impersonators, stalkers, spammers and phishers. The attacker's goal is to collect private user account information from a social networking site. We assume an attacker may build or use tools to automate many phases of its attacks. In our model, we also consider the attacker's need for anonymity. For this, we assume an attacker can create and use multiple fake accounts (that do not provide truthful profile information) and may optionally use hijacked or public machines. Thus, in this case, a fake profile is defined to be a profile in which the personal information is vastly missing or different from that of the person owning the profile. This would allow the attacker to maintain its anonymity even in the event that Facebook, upon detecting such stalking activities, would attempt to correlate the fake account's identity to the IP address of the machine used to open the account.

Fake Locations Attackers may try to run CheckIn for venues where they are not present, try to feed fake location information to other clients, and in general attempt to fraudulously obtain private information from other clients.

3 Related Work

3.1 Infiltration Attacks

Bilge et al. [12], study solutions for collecting personal user profiles from various OSNs, including Facebook. They conjecture and prove that people are more willing to accept friend requests from people they already know. They devise an impersonation attack which they test on 700 users.

Caverlee and Webb [13] conducted a large scale study on MySpace, where they discovered interesting patterns, such as high account abandonment rates, language/location correlations and interestingly, that privacy is becoming a concern factor even and mostly for younger users. Nazir et al. [14] conducted a similar study on Facebook, through the development and deployment of three applications that gained significant popularity. The focus of their study is on behavior within a community—in their case, the communities adopting their applications.

Another MySpace study was conducted by Webb et al. [15], to study social spamming. The concept of social honeypots is introduced, which are MySpace accounts created specifically for attracting spam. The results show that social spammers exhibit temporal and geographic patterns, which may be used to automatically detect and even eliminate them. Caverlee et al. [16] continue this work with the proposal of a trust establishment solution for social networks. Trust between two users is defined to be a factor of the quality of the interaction between the two users.

Boshmaf et al. [17] have shown that OSNs are highly vulnerable to infiltration attacks. Recent studies [18] also suggest that the currently deployed Facebook immune system is not sufficient for preventing this class of vulnerabilities.

Our Differences Previous work has identified various infiltration problems and proposed solutions that preserve the privacy of users from curious providers and other users. Our work differs in the magnitude of the study and in the novel, location based defenses we devise.

3.2 Securing Social Networking Accounts

Our work builds on recent work on preserving the privacy of users from social network providers. Tootoonchian et al. [19] devised Lockr, a system for improving the privacy of users through the concept of social attestations—credentials proving a social relationship. Luo et al. [20] proposed FaceCloak, that provides fake information to the social network and stores the account information encrypted on a different server. Baden et al. [21] introduced Persona, a distributed social network with distributed account data storage. Sun et al. [22] proposed a similar solution, extended with revocation capabilities through the use of broadcast encryption.

Puttaswamy and Zhao [23] argue that untrusted providers should be allowed to store only encrypted data, and move the application functionality to client devices. Users store "friendship" and "transaction" proofs on the provider site, cryptographically encrypted tokens encoding friend relations, location-centric reviews, etc. This approach supports a wide range of location-based applications, including collaborative content downloading, social recommendations, or co-location events. Our goal is to protect user privacy both from providers *and* from other users. Account privacy does not prevent users from accepting invitations from fake, Sybil accounts.

3.3 Location Verification

Most of initial work on location verification focuses on fine grained localization. In ad hoc networks capable of both RF and ultrasound communications, Sastry et al. [24] introduced the ECHO protocol. In ECHO, location claims are verified by selecting multiple nodes within the transmission range of the prover. Each verifier sends a packet to the prover over RF which the prover must echo over ultrasound. Howard et al. [25] study the use of indoor Wi-Fi to localize moving robots. Chiang et al. [26] propose a distance bounding approach for solving the location verification problem in an ad hoc network that contains multiple verifiers. The solution relies on a multilateration technique, where the prover must respond simultaneously to challenges issued by multiple verifiers situated in its vicinity. We note that while these solution could be applied in the context of GSNs, our work imposes significantly lower costs on venues and does not rely on the existence of widely distributed, non-colluding, altruistic verifiers.

Zhu and Cao [27], through their APPLAUS system, took the next step, by proposing an approach where co-located devices cooperate to build location proofs over Bluetooth. This approach is however not ideally suited for geosocial networks. The ease of creating Sybil accounts can allow attackers to thwart this defense: multiple accounts controlled by the attacker can claim to be co-located. If the Sybil accounts form more than $(n-1)/3$ of all the n devices claiming to be at a location L, consensus cannot be achieved by the honest users at L [28]. Furthermore, when the number of Sybil accounts exceeds $2h+1$, where h is the number of honest users located at L, the Sybil accounts can reach consensus, (fraudulently) establish their presence at L and frame the honest users as Sybils.

Saroiu and Wolman [29] explored the location proof concept—a piece of data that certifies a receiver to a geographical location. The solution relies on enhanced access points (APs), able to issue such signed proofs. Such APs add their location to their beacons. Upon client request, the APs issue signed location certificates, containing the client's identity, the AP's identity, location and timestamp. Note that multiple certificates can be combined to triangulate clients and obtain more accurate positions. Luo and Hengartner [30] extend this concept with client privacy, achieved with the price of requiring three independent trusted entities. Access points seem an ideal candidate for solving the secure location verification problem: they are widely accepted and the solution implies relatively small changes to their code base. They do however also have disadvantages: Most APs are owned by regular users who lack the incentives to install new code. Moreover, AP owners may find it profitable to provide (even proactively) fake location certificates.

3.4 Visual Security Indicators

Previous work has studied the problem of introducing visual indicators to warn users about possible cyber security risks. Egelman et al. [31] examined the ability of active and passive warnings included in web browsers to convey the risks of phishing attacks to users. They performed a user study and compared the effectiveness of warnings based on an information processing model [32]. The conclusions of the study are that the effectiveness of a warning message depends on two factors, (i) hazard matching, which depends on how accurate the warning message is in conveying the risk and (ii) arousal strength, that defines how the users perceive the warning and if they are motivated to prevent the risk.

Shin and Lopes [33] proposed a set of visual indicators, inspired by traffic lights, in order to secure users against the SSLstripping attack. They assign traffic colors to the text fields in a webpage based on the output of an attack detection mechanism. The green color represents a secure webpage, red represents an insecure one, and yellow denotes the inability of the detection mechanism to make a decision. These visual indicators raise the awareness of users to the security of visited pages. Care needs to be taken to avoid annoying users with hard to parse web pages.

4 Social Network Infiltrations

In this section we show that an attacker can obtain private information of targeted victims by infiltrating OSNs, using Sybils—fake accounts controlled by the attacker.

4.1 Attacks

We identify several attacks that can be launched against social networking users and formally derive a new infiltration attack called the *3-Clique* attack. Let M be the attacker, using a fake account to hide its identity. The goal of the attacks is for M to get access to a victim A's account data (mainly its profile information). That is, M wants to change its relationship with A, such that it has access to A's profile data. We assume that initially M has no access permissions to A's profile data. As we show

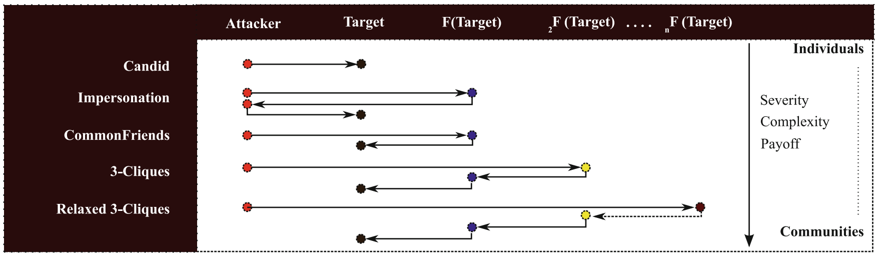


Fig. 1 Attack Map: While the complexity of the attack increases with the attacks in the lower half, the payoff is much higher too

in Sect. 4.2, if an attack fails, M can generate a new fake account and perform other refined attacks (Fig. 1).

Candid Attack M issues a friend invitation to A. If A issues an accept, A and M become friends, that is, A gives explicit rights to M to its profile data. Therefore, M learns A's profile.

Impersonation Attack Deciding the authenticity of profile information is a hard problem. This attacks leverages this observation: M chooses a friend F of A and copies its profile, making it its own. M sends an invite to A with a message of the format "*I have lost my old account and made a new one. Please accept this request.*". Since this behavior can be legitimate, A accepts the invite and reveals its profile. A similar attack was first described in [12].

Chameleon Attack Trust is often based on familiarity. This attack builds upon the hypothesis that users tend to accept invitations more easily when they are sent by people with whom they share mutual friends. Then, M initially collects the set of friends of A and issues invitations to everyone or a selected subset (see the *3-Clique* attack discussed next). M then waits for the first of two events: (i) a desired percentage of invitees issue an accept or (ii) a pre-determined time interval lapses (two days in our experiments proved to be sufficient, as shown in Fig. 6b). M continues with the *candid* attack (against its original target, A). At this point M will have a higher chance of succeeding in becoming A's friend: M and A will likely share mutual friends.

The 3-Clique Attack Community infiltration in the context of social networks is a type of attack where an attacker targets individuals who are connected together in the form of a community. They could be leveraged for stealing information that belongs to a group to launching an *association fallacy* [34] against the entire group. An *association fallacy* is an inductive informal fallacy which asserts that qualities of one thing are inherently qualities of another, merely by an irrelevant association. For instance, with enough attackers infiltrating a community, the collective qualities that represent it can be altered. In the following, we formally define the *3-Clique* attack which can be used to launch such association fallacies.

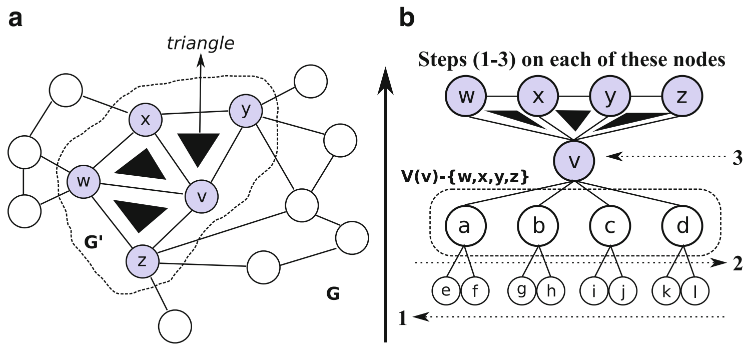


Fig. 2 (**a**) **Infiltrating a network**: Node v has the highest $\delta(v)$ in G', i.e., the attacker's payoff is higher if it establishes a link with v. (**b**) **The *3-Clique* Attack**: Infiltration is done in the decreasing order of δ. To establish a link with v (highest δ value)—recursively link with the *n*th hop network of v in the increasing order of *Social Closeness* to the level above

Let a *friendship 3-clique* $\Delta = (V_\Delta, E_\Delta)$ of a graph G = (V,E) denote a subgraph such that $V_\Delta = \{u, v, w\} \subset V$ and $E_\Delta = \{(u,v), (v,w), (w,u)\} \subset E$. Then, let $\delta(v)$ denote the number of friendship 3-cliques of user v. Let the *social closeness* metric, $SC(u,v) = \frac{|F_u \cap F_v|}{|F_u|}$ denote the ratio of friends of u. We define then the *friendship weight* of the link between users u and v to be $w(A,B) = \frac{SC(A,B)+SC(B,A)}{2}$. Intuitively, this is taking into account two factors: what proportion of A's friends are also B's friends and what proportion of B's friends are also A's friends.

The *3-Clique* attack is executed using Algorithm 1 and is shown in Fig. 2b. Let $G' \in G$ denote a subset of the OSN, a tightly knit community that the attacker targets (*line 1*). For each member $v \in G'$, the attacker computes $\delta(v)$ (*line 2*), the *3-cliques* of user v using the method given in (*lines 15–27*). Let v be the member of G' with the highest δ value (*line 4*). The attacker computes the first hop network of v, excluding all the users in G'(*line 5*). Then, the users in this network are ordered decreasingly on the value of their friendship weight to v (*line 6*). An invitation is sent to the second hop network of v (friend-of-friend network) who are again ordered based on their social closeness to each friend of v (*lines 7–11*). Then, after a delay period T, the attacker sends invitations to the first hop network of v (*line 12*). The delay is used to allow the invited second hop network members to accept the invitations. Finally, after another delay period T, the attacker invites the target v (*line 13*). The above process is repeated for all members of G', in decreasing order of their δ values. This is based on the observation that users with high δ values are socially tied to a higher number of groups. Such users may not only be more willing to accept random invitations but more importantly, establishing a friend link with them may further influence other members of G' into accepting the attacker's invitation. An extension of the attack is what we call the *Relaxed 3-Clique* attack where the attacker can start from the *n*th hop network of a victim instead of the 2nd

Algorithm 1 Enhanced Infiltration using 3-Cliques

```
1. $G' = Sample(G)$
2. $3CSet = get3Cliques(G')$
3. DO
4.     $v = popUserWithMax3Cliques(3CSet)$
5.     $firstHopNet(v) = v.friends() - \{x : x \in G'\}$
6.     $Sort(user \in firstHopNet(v), w(v, user), DESC))$
7.     FOREACH friend IN firstHopNet(v) :
8.         $secondHopNet(v) = friend.friends()$
9.         $Sort(user \in SecondHopNet(v), w(friend, user), DESC)$
10.        inviteAll(SecondHopNet(v))
11.    END FOR
12.    Schedule(inviteAll(FirstHopNet(v)), T)
13.    Schedule(invite(v), 2T)
14. WHILE(len(3CSet) > 0)

15. PROCEDURE $get3Cliques(G' = (V, E))$ :
16.    Setcliques = newSet() :
17.    Construct Friends(user) $\forall$ user $\in$ V
18.    FOREACH relationship IN E :
19.        //*relationship consists of* $(user_1, user_2)$
20.        $user_{min} = min(|user_1.friends()|, |user_2.friends()|)$
21.        FOREACH user IN $user_{min}$.friends() :
22.              $IF(user \in (relationship - \{user_{min}\}).friends())$
23.                store $\{user, user_1, user_2\}$
24.              END IF
25.        END FOR
26.    END FOR
27. END PROCEDURE
```

hop network as we demonstrated. However, due to space constraints, we will not be discussing this attack further but the attack plan itself is shown in Fig. 1.

4.2 *Infiltration Evaluation*

Here we provide a brief overview on the architecture of iFriendU, our attack system, and then describe the results of our infiltration attempts using this system.

iFriendU Architectural Overview Our prototype attack system[2] relies on an *attack plan* prepared for each of the attacks discussed in Sect. 4.1 and is illustrated in Fig. 3. The *crawler* component is responsible for crawling the target social networking site and collecting information on users as a seed for the attacks. As

[2]Please note that our attack system *does not* collect any personal information of users nor does it send any malicious data to users.

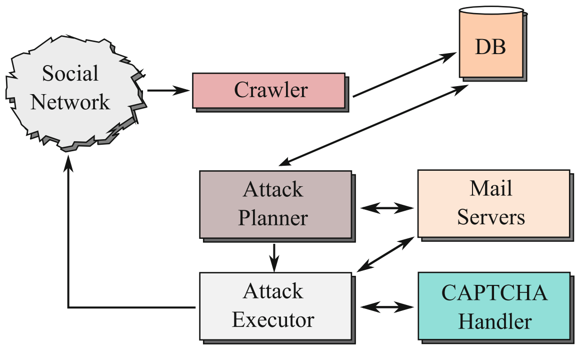


Fig. 3 An architectural overview of iFriendU. The crawler module extracts initial, publicly available information about social network users, including intended targets. The attack planner and executor modules rely on existing mail servers and CAPTCHA solving tools to defeat the defenses used by social networks and send invitations to intended targets

Facebook allows anyone to view an arbitrary user's friends list in most cases, we provided our *crawler* with a seed Facebook account using which it recursively crawls the entire network limited by a *depth* parameter customizable through code. We used a 2.4 GHz Intel Pentium 4 with 2 GB RAM to complete crawling of about 50,000 users in less than 5 h.

The *attack planner* relies on the attack plan (see Fig. 1), which is a concise-representation of an attack. For instance, for a *3-Clique* attack, we can specify the order in which the attack execution takes place using "$COM(CLIQUES\ ASC) - L1(SC\ DESC) - L2(SC\ DESC)$" which means, first arrange the the nodes (belonging to the target community) based on their *3-Clique* value. For each node in this order, arrange its 1st hop network in the descending order of the social closeness of each node and so on. The *attack planner* also prepares a list of fake accounts (needed to preserve sender anonymity) and a set of targets listed along with their friends and proceeds to computing the 3-Cliques by interfacing with a backend MySQL database using Algorithm 1.

The plan generated by the *attack planner* is used as input by the *attack executor*. We have implemented the *attack executor* as a backend driver using Java for Firefox. The *attack executor* launches the browser with itself as the proxy server and then injects Javascript to execute the *attack plan*. The *attack executor* uses fake accounts to send friend invitations to the target accounts included in the plan. If during its operation the *attack executor* encounters a CAPTCHA, it hands it off to the *CAPTCHA Handler* which uses automated CAPTCHA solvers (e.g., CaptchaBuster [35]) to solve the them. If all attempts fail, the component sends us an email requesting a manual inspection of the CAPTCHA.

At this point, it is important to mention that the *attack executor* waits for a specific time interval between sending friend requests. The reason for waiting between sending successive invites is that Facebook suspends the account if it sends too many invites. Figure 6a shows the results of our experiment when changing the inter-invitation delay time from a *few hundred ms* to *100s*. The *y*-axis shows the number of invites successfully sent before being banned by Facebook. Note that this number grows quickly but saturates. Our conjecture is that Facebook forbids an account to send more invites when its number of pending invites (sent but not yet answered) exceeds a given value (between 400–500). Because our experiments were designed around sending 1000 s of invitations, we extended the *attack executor* to adapt its sending rate depending on its current state. For the first 500 invites, we chose the inter-invite delay randomly between 1 and 15 s. For the next 500 invites, we increased the upper limit of the delay to 60 s. For the remaining invites, the upper delay limit was further increased to 100 s. We were then able to consistently send more than 1500 invites from an account in only a few hours. Note that a fake account can only be used to send a limited number of invites, since the number of pending invites *will* at some point exceeds Facebook's limit.

On a related note, since the fake accounts used by the *attack executor* need to be validated, we setup our own mail server. Both the *attack planner* and *attack executor* interface with the *mail server* for creating accounts as required or for confirming friend requests from other users.

We have launched the following attacks using a collected relationship graph with 178,000 Facebook users and 339,000 friendship relations.

Chameleon Attack We illustrate the *Chameleon* attack we implemented, using Fig. 4. In the first stage, we selected a random set of 1577 Facebook accounts from our crawled data and launched a *candid* attack using a fake account. Figure 5a shows the result of this experiment two weeks after sending the invites: 742 users accepted the invitations, 309 rejected it and 526 were still undecided (pending). From the 309 users who rejected us, we selected a random set of 72 users and sent invitations to all their friends. In order to avoid detection by Facebook, we refrained from sending mass invitations to the friends of these 72 users. Instead, we attached them to nine different fake accounts, each dealing with 8 out of the 72 users: each fake account

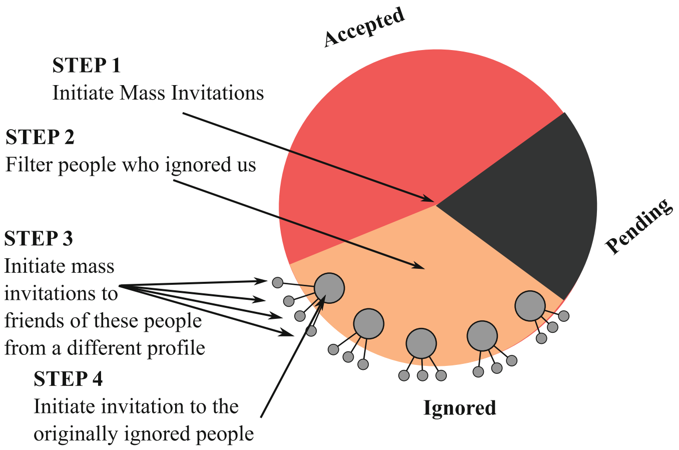

Accepted
STEP 1
Initiate Mass Invitations
STEP 2
Filter people who ignored us
STEP 3
Initiate mass invitations to friends of these people from a different profile
STEP 4
Initiate invitation to the originally ignored people
Pending
Ignored

Fig. 4 *Chameleon* Attack: Attempt to obtain as many profiles as possible from a set of select targets. The attack attempts to first befriend the friends of the targets, before befriending the targets. The common friends achieved by this process improve the acceptance rates of the invitations by the targets

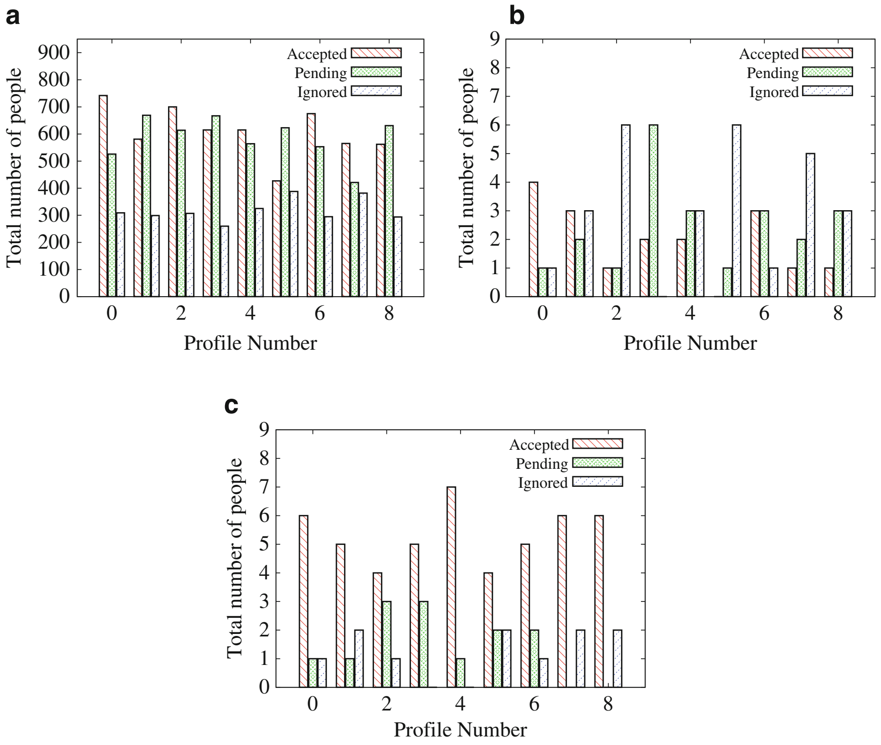


Fig. 5 (**a**) Chameleon Attack Evaluation: Results from the first stage of invitations to the 1577 users (**b**) Chameleon Attack Evaluation: Results from the second stage of invitations to the 72 users (**c**) *3-Clique* Attack Evaluation: Last stage of the *3-Clique* attack against the 150, 3-clique members of the 72 target users

is responsible for sending invitations to the friends of 8 of the 72 targeted users ($\sim$ 1600 invites per fake account).

Finally, we use these nine accounts to complete the *chameleon* attack, by launching a subsequent *candid* attack against the eight users attached to each of the nine fake accounts. Figure 5b shows the result of the *chameleon* attack, again grouped by associated fake account. In total, about 41.7 % of the users fell for the attack while 32.8 % sustained it. The rest of the 25 % were still undecided. Ironically, when we re-sent invitations to the people who rejected us one week later, several have accepted us which indicates that persistence pays off. This is due to a Facebook security glitch that we discovered: flagging a user as *unknown* has no effect. That is, even after being rejected, an attacker can re-send the invite any number of times and it will always be shown as a fresh invite to the victim.

3-Cliques Attack We have run Algorithm 1 on the relationship graph collected from Facebook (178,000 users and 339,000 friendship relations) and discovered 679,000 3-cliques in less than five minutes (had a similar performance on a different dataset as well [36]). Note that the large number of 3-cliques follows from the small world property of OSNs [37]. To test our *3-Cliques* attack, we targeted the same 72 users used in the *chameleon* attack but after a period of 60 days. Each of the 72 users has the value δ smaller than 500. This is typical for Facebook users, who usually have fewer than 130 friends. In fact, the total number of 3-clique friends for these 72 users was 150 (note that some users participated in multiple *3-Cliques*). We executed the steps specified in the *3-Cliques* attack description (see Sect. 4.1 against these 150 users (which form the target community G‘). Figure 5c shows the result of the *Chameleon* attack, again grouped by associated fake account. The acceptance rate was *79 %*. This shows that the *3-Clique* attack is 75 % more efficient than the *Chameleon* attack.

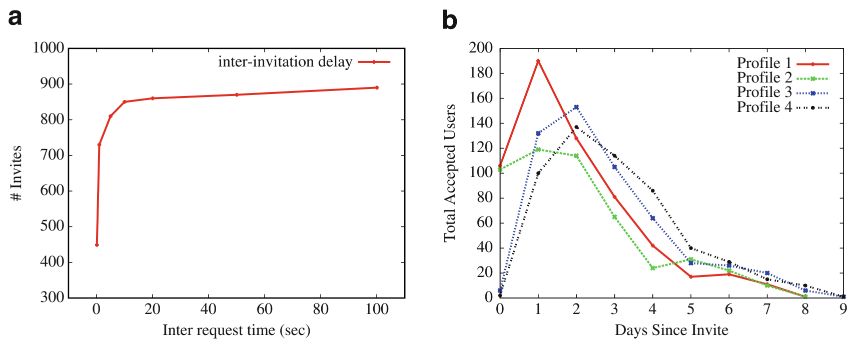
Additional Statistics Following our *Chameleon* and *3-Clique* attacks, we have monitored several variables. First, we have measured the number of invites accepted per day, following the beginning of each invite. Figure 6b shows our results for four out of the nine fake accounts used to send invites. Most invites are being accepted in the first three days after the beginning of the experiment.

Another metric of interest is the distribution of the number of friends per Facebook account. Figure 6c shows the average over the 10,000 different accounts targeted in our experiments, split over three categories: the accounts that accepted, rejected or decided to keep it pending. In the active and rejected groups, most accounts have between 50 and 450 friends. The pending group is more interesting, since most users have fewer than 200 friends. In particular, there are almost 600 users with less than ten friends. One explanation for this distribution is that some of the accounts that have neither accepted nor rejected our invitations may be inactive. These users have tested Facebook briefly but are not active.

5 Infiltration Attack Defenses

We propose to use location information certified by geosocial networks, to validate invitations received from other users. Location information, in the form of check-ins, enables users to prove co-location with prospective friends.

One important question is whether geosocial networks register sufficient numbers of check-ins for them to be a relevant defense tool. To answer this question we have collected publicly available data from 781,239 active Foursquare users. To obtain this data, we have relied on the Foursquare API for querying the profile of users and leverage Amazon EC2 for issuing parallel queries. An essential step in crawling was to maintain a queue of user IDs that need to be crawled. To efficiently parallelize our task, we leveraged Amazon's queueing service called SQS to distribute tasks to various computing instances. The cost of using SQS is dependent on the number of messages that transfer through the system and the data traffic that is involved. In total, we utilized 70 Amazon micro instances for a period of 2 weeks, to rate limit our requests made to the Foursquare servers.



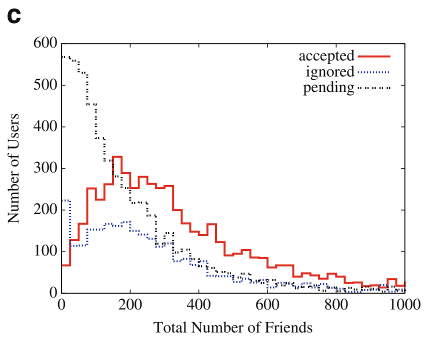
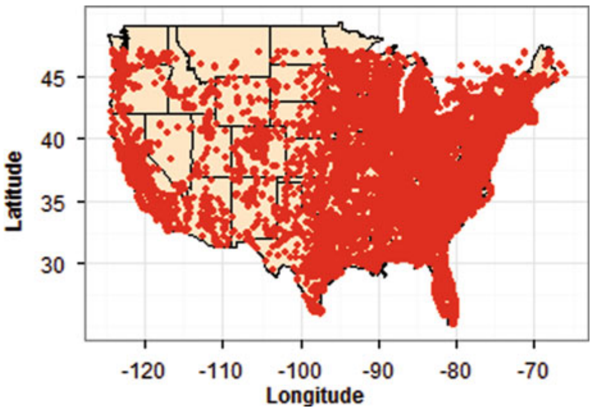


Fig. 6 Attack Statistics: (**a**) Capping invitations as a function of inter-invitation delay times. (**b**) Accept rate timeline. Note that the first two days following the invitation are instrumental on the acceptance rate. (**c**) Friend count distributions of accepted, rejected and pending user accounts. The accepting users tend to have more friends than those who ignore random invitations

Fig. 7 Geographical distribution of Foursquare users: Foursquare is most popular in the eastern half of the United States with New York being the most popular city



For each user, the collected data consists of the user profile, the total number of friends, check-ins and "days out" (days the user was actively performing check-ins). Figure 7 shows the geographical distribution of the user "home cities" in the US subset of the Foursquare dataset. It proves that not only the coasts but also the entire eastern half of the US is actively using Foursquare. These results are in agreement with results from Cheng et al. [38] and Cha et al. [39].

We propose to build the solution on a distributed, peer-to-peer online social network: Each client locally stores and maintains its check-in history—the trace of (location,time) pairs certified by the social network. This approach builds on existing work, and enables social network users to preserve their privacy from curious providers.

Let H_A denote the check-in history of client A. For each supported venue V and during each epoch e, the provider S generates a random *presence token*, $Tk_{V,e}$. $Tk_{V,e}$ is only provided to clients that successfully check-in at V during epoch e. S also maintains for venue V a *vicinity set* $\phi_{V,e}$, containing the presence tokens for a predefined set of neighboring venues. $Tk_{V,e}$ and $\phi_{V,e}$ change once per epoch for each venue.

For each venue V, we define the set $\bar{V} = \{V_1, .., V_a\}$, where V_1 denotes the geo-coordinates of V and $V_i \in V_{i+1}$, $\forall i = 1..a - 1$ are areas containing the location of V, sorted in decreasing order of accuracy. For instance, the areas in $\bar{V}$ can be defined as concentric squares of predetermined dimensions, centered at V or can range from V's exact street address to only providing V's city (or zipcode, country, etc). Similarly, for a time T, we define the set $\bar{T}$, of t intervals $T_1, .., T_t$, all containing T and ordered with decreasing levels of accuracy (e.g., hour, epoch, day, month, year).

We define the infiltration attack detection solution to be a set of protocols InfiltrationCheck = { Setup, VenueMaintenance, CheckIn, TopoProof, TopoMatch }. Setup is run initially to generate system-wide parameters. The VenueMaintenance procedure is executed periodically by the provider S for each registered site. A client invokes the CheckIn procedure when (i) the user explicitly wishes to perform a check-in, or (ii) *seamlessly*, when it detects to be located around a venue of interest. A successful CheckIn enables the client to collect a token that uniquely identifies the place and time of the operation.

During the invitation protocol, inviters reveal a function of their check-in tokens, allowing invitees to perform the following private location based operations: (i) verify the location claims made by the inviter in its profile (e.g., past locations, schools attended, jobs, etc) and (ii) determine the location affinity between the user and the inviter, that is, how many common places have they visited simultaneously.

TopoProof allows the inviter to prove his past location claims—(location, time) pairs—with a desired accuracy level. Besides allowing inviters to prove claims of the type "we met last year at a conference", it also enables a curriculum verification protocol: claimed studies, employment, vacations can be corroborated by existing check-ins. TopoMatch allows the invitee to privately infer her *location affinity* with the inviter. We define the location affinity of two users to be the size of the intersection of the sets of locations (and times) frequented by both users.

The user is notified if the results of TopoProof and TopoMatch are not conclusive or are negative. The user then needs to decide whether it will trust a user with no common location context as a friend.

In the following, we describe each of the above procedures.

Setup(): S generates a public/private key pair and distributes the public key to all clients.
VenueMaintenance(S()): At the beginning of each epoch e, for each registered venue V, S generates a fresh random presence token, $Tk_{V,e}$, then updates the vicinity set $\phi_{V,e}$ to contain the presence tokens of all of V's neighboring venues.
CheckIn($C(Id, V, T, H_C, pub_S), S(priv_S, V)$): Executed between a client C with check-in history H_C, located at venue V at time T and the provider S. C runs one of the location verification protocols from Sect. 6, to prove its presence at V. If it fails, S aborts. Otherwise, C generates a fresh random key k, generates $E_k(Id(C))$ and sends it, along with V and T to S. S performs the following four actions.

- Generate a fresh key k_a and associated hash chain $w_i = H^i(k_a)$, $i = 1..a$ and a fresh key k_t and associated hash chain $\theta_j = H^j(k_t), j = 1..t$.
- Given the set $\bar{V}$ of areas containing V, with decreasing accuracy levels, generate the set $\mathscr{E}_V = \{E_{w_i}(V_i)|i = 1..a\}$, containing the symmetric encryption of all the V_i's with key w_i. Given the set $\bar{T}$ of time intervals containing T, with decreasing levels of precision, similarly generate the set $\mathscr{E}_T = \{E_{\theta_j}(T_j)|j = 1..t\}$.
- Generate signature $\sigma_{V,T} = \{S_S(E_k(Id(C)), \mathscr{E}_V, \mathscr{E}_T)\}$.
- Send $\sigma_{V,T}$, the keys k_a and k_t, along with the presence token $Tk_{V,e}$ and vicinity set $\phi_{V,e}$ of venue V during the current epoch e ($T \in e$) to C. C records in her history set H_C the following tuple: $(V, T, Tk_{V,e}, \phi_{V,e}, \bar{V}, \bar{T}, \mathscr{E}_V, \mathscr{E}_T, k, k_a, k_t, \sigma_{V,T})$.

TopoProof($B(H_B), A()$): The inviter B sends to the invitee A a list of locations claimed in his profile (e.g., locations where studies were done, current work place, current city of residence). If insufficient such information is provided, TopoProof returns 0 ("reject"). Otherwise, A and B negotiate the level of accuracy desired when revealing B's past locations. Let $\alpha \in \{1..a\}$ and $\tau \in \{1..t\}$ be the chosen spatial and temporal accuracy levels. To prove presence within an area V_p within a time interval T_p, B retrieves from H_B all the matching entries. The search is performed on the sets $\bar{V}$ and $\bar{T}$ of each entry in H_B. Let $(V, T, Tk_{V,e}, \phi_{V,e}, \bar{V}, \bar{T}, \mathscr{E}_V, \mathscr{E}_T, k, k_a, k_t, \sigma_{V,T})$ be a matching entry. That is, $V_\alpha \in \bar{V}$ and $V_\alpha \subseteq V_p$ while $T_\tau \in \bar{T}$ and $T_\tau \subseteq T_p$. B computes $w_\alpha = H^\alpha(k_a)$ and $\theta_\tau = H^\tau(k_t)$. B sends $V_p, T_p, \mathscr{E}_V, \mathscr{E}_T, k, w_\alpha, \theta_\tau, \sigma_{V,T}$ to A. A performs the following verifications:

- Use $\sigma_{V,T} = \{S_S(E_k(Id(B)), \mathscr{E}_V, \mathscr{E}_T)\}$ and the key k, to verify that S's signature binds B to the sets $\mathscr{E}_V$ and $\mathscr{E}_T$.
- Use the key w_α to decrypt the entry α from $\mathscr{E}_V$ and verify that the resulting $V_\alpha \subseteq V_p$. Use the key θ_τ to decrypt the entry τ from $\mathscr{E}_V$ and verify that the resulting $T_\tau \subseteq T_p$.
- If either verification fails, return 0.

TopoMatch$(B(H_B, max_r), A(H_A, k, max_r))$: TopoMatch is initiated by B to allow A to privately derive the number of check-ins performed in the vicinity (both space and time) of check-ins of B. max_r is a system parameter, used for hiding the length of check-in history sets. The following steps are executed:

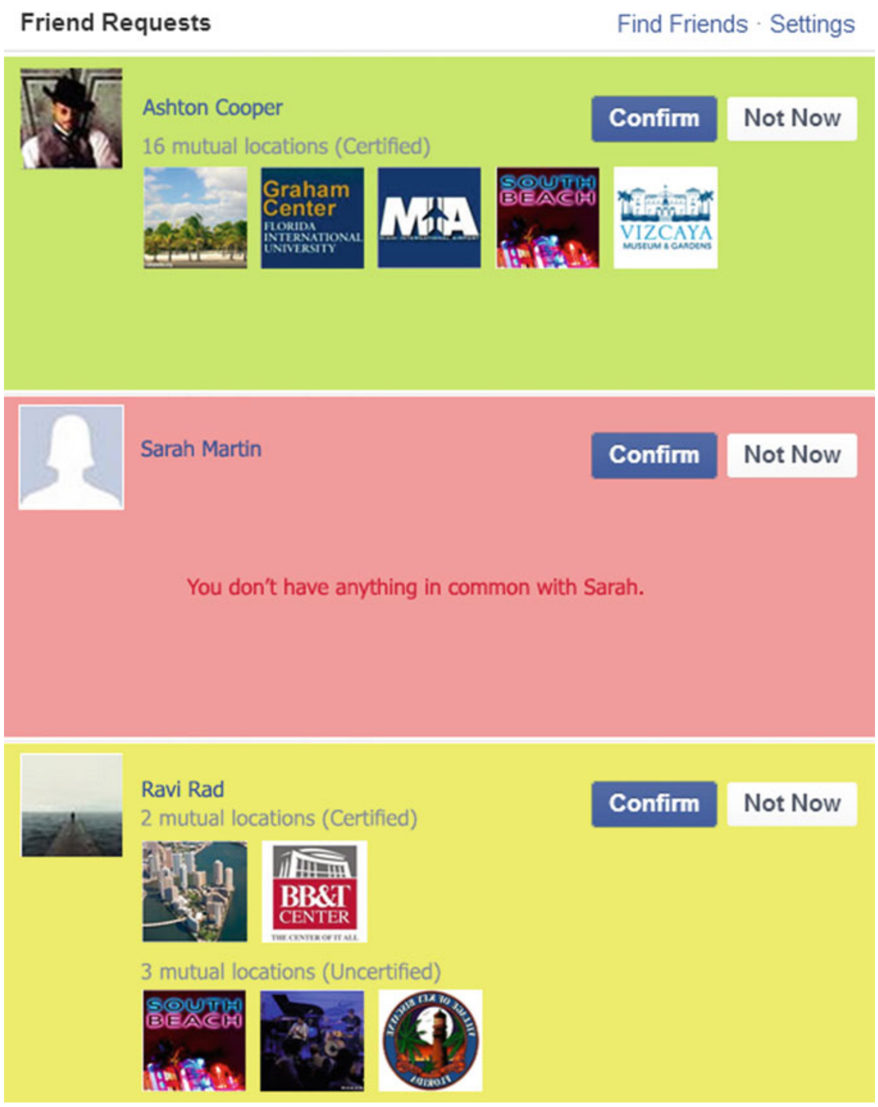
- A and B agree on a random blinding factor u (using a pre-commitment step). B initializes an empty set P.
- For each entry in the check-in history H_B corresponding to a venue V, B retrieves the vicinity set $\phi_{V,e}$, for the epoch e when the check-in took place. For each token $Tk_{V',e} \in \phi_{V,e}$ corresponding to a venue V' in the vicinity of V, B computes the hash $H(u; Tk_{V',e})$ and inserts it in the set P.
- B generates a value r uniformly at random from $[1..max_r]$. If P has less then r elements, B generates $r - |P|$ random values (of the same bit size as the output of the hash function H) and adds them to P. B computes a random permutation π and sends the permuted set $\pi(P)$ to A.

A then initializes a counter $ctr = 0$. For each entry in the check-in history H_A, for a venue V during epoch e, A retrieves the presence token $Tk_{V,e}$. A computes $H(u; Tk_{V,e})$. If $H(u; Tk_{V,e}) \in \pi(P)$, A increments ctr. TopoMatch returns ctr. The value r and the padding in the last step are used to hide the number of check-ins of B from A.

5.1 Taking Action: Infiltration Warnings

The experiments in Sect. 4 show that users are often unaware of the security risks associated with their actions. While TopoProof and TopoMatch can detect invitations received from people with no common context, an important challenge remains to communicate this information to users in an effective manner. This is important, as empirical studies have shown the inefficiency of security mechanisms when usability is relinquished [40].

We propose the use of visual indicators as proposed in [33], to convey the output of TopoProof and TopoMatch to users. First, we propose to extend the friend request notification in social networks with information provided by TopoProof and TopoMatch. For instance, Fig. 8 depicts an extended friend request list in Facebook. The output of TopoProof and TopoMatch is presented to the user, on a background corresponding to the inferred safety of the inviter. The color, ranging from green (safe) to red (unsafe), is decided based on the output of TopoProof and TopoMatch. If the user shares more than a threshold number of TopoProof certified locations with the inviter, the color is green. Otherwise, the difference between yellow and red is made based on a second threshold of the number of certified locations shared with the inviter.



Ashton Cooper
16 mutual locations (Certified)

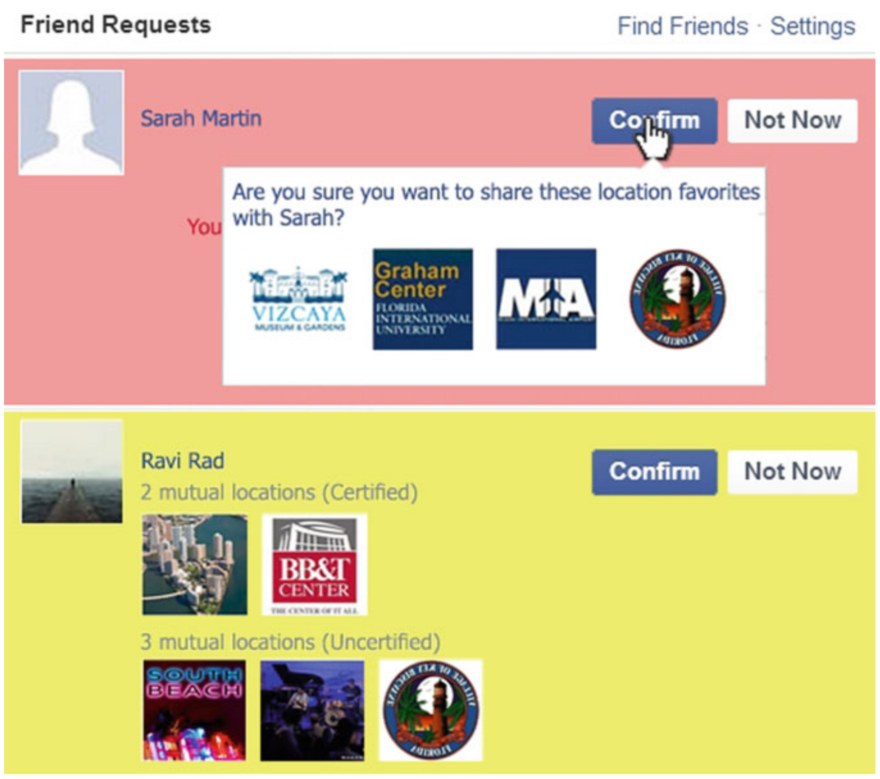
Sarah Martin

You don't have anything in common with Sarah.

Ravi Rad
2 mutual locations (Certified)
3 mutual locations (Uncertified)

Fig. 8 Visual indicator for pending invitations. Background color denotes the safety of befriending an inviter, as inferred by TopoProof and TopoMatch

If the user decides to accept a "green" inviter, the process takes place as usual in Facebook. If the user decides to accept a "yellow" or "red" inviter, we extend the Facebook interface with a second visual notification. Figure 9 illustrates the notification displayed, consisting of a sample of locations from the user's past history, that would be shared by default with the inviter. If the user is comfortable with this choice, the invitation is accepted.



Are you sure you want to share these location favorites with Sarah?

2 mutual locations (Certified)

3 mutual locations (Uncertified)

Fig. 9 Visual indicator displayed when the user decides to accept a suspicious inviter. The illustration extends the example shown in Fig. 8. The user needs to decide whether she is comfortable sharing her past history of preferred locations with this inviter

6 Preventing Location Fraud

TopoProof and TopoMatch rely on certified user location information. Geosocial networks use various incentives, including financial, to encourage users to perform check-ins, thus report their location information. The use of incentives introduces reasons for cheating, motivating users to commit *location fraud*: falsely claim to be at a location, to receive undeserved rewards or social status. Even with GPS verification mechanisms in place, committing location fraud has been largely simplified by the recent emergence of specialized applications for the most popular mobile ecosystems (LocationSpoofer [6] for iPhone and GPSCheat [7] for Android).[3] Such behavior places undue burden on participating venues, as proved by the recent surge in the numbers of fake check-ins and "*instant*" mayors [41].

[3] In fact, He et al. [8] proved the feasibility of fake check-ins in Foursquare.

To address this problem, we exploit the insight that venues have the most to gain from properly rewarding users—their main goal is to retain customers and attract new users. We introduce then a suite of venue-oriented, secure location verification mechanisms, that require participating venues to deploy minimalist equipment. To promote their adoptability, we design the solutions to be not only secure and correct, but also user friendly, economical and easy to deploy. They consists of mechanisms that (i) broadcast unpredictable Wi-Fi SSIDs, (ii) display QR codes encoding venue certified information, and (iii) implement challenge/response protocols.

6.1 Requirements

A location verification solution needs to satisfy several requirements. First, *correctness*, ensuring that a check-in at a venue V succeeds with high probability if the client is located at V. Second, it should provide *check-in security*, preventing clients from performing check-in fraud. Furthermore, the solution should be *user friendly*, imposing a small overhead on its users. The solution should also minimize the investment cost for the social network provider, participating venues and users.

6.2 The Solutions

We propose here a novel approach, relying on two components of the geosocial network ecosystem: the venues and their owners. One insight is that venues *know* the ground truth, being at the center of locations claimed during check-ins. Moreover, venue owners have incentives to stop location fraud, being motivated to reward the proper frequent customers. We require then participating owners to install additional equipment within their venues.

The equipment should be inexpensive, portable and should not require Internet connectivity. While we have evaluated our solutions on simple smartphones, we note that recent technologies, such as Apple's iBeacon [42] and TI's SensorTag [43] are not only significantly cheaper (e.g., a SensorTag costs $25) but are expected to have a major impact on the indoor localization market, including shopping malls and restaurants [44].

We propose two location verification mechanisms: a (i) **Feedback-enabled Embedded System** (FES) equipped with an LCD screen and a proximity sensor or a (ii) **Network Embedded System** (NES) capable of communicating with nearby devices. Most smartphones have *all* these three capabilities and can be used as an implementation platform. In the following we use $\text{S}{\small\text{POTR}}_V$ to refer to the device that a venue V deploys within its site to "*exact*" this mechanism. We also consider time epochs to be further divided into frames. The length of time frames is a system parameter and should be on the order of seconds. Algorithm 2 shows the pseudocode of our solutions.

Algorithm 2 Pseudocode of FES and NES that runs on Spotr$_V$, the device installed at a venue V

```
1.Object implementation SPOTR_V;
2.  int T;               #timestamp in ms
3.  int ΔT;              #expiration interval
4.  int pk_V, pr_V;      #signature key pair of deviceat V

5.  Operation main()
6.      T := getCurrentTime();
7.      if (algoType = FES) then
8.          ff := sign(pr_V, T, ΔT);
9.          string qrCode := T + ΔT + ff;
10.         displayCode(qrCode);
11.     if (algoType = NES) then
12.         Connection con := waitForConnection();
13.         con.send(T, ΔT, R);
14.         int T_1 := getCurrentTime();
15.         string msg := con.recv();
16.         int T_2 := getCurrentTime();
17.         if (T_2 − T_1 < threshold) then
18.             ff := sign(pr_V, msg);
19.             con.send(ff);
20.     fi fi
21. end
```

We consider first a communication restricted equipment, where a local communication interface (e.g., Wi-Fi) is not available. The equipment at venue V, which we denote by SPOTR$_V$, verifies the presence of a user by making use only of its display, cryptographic signatures and Quick Response Codes (QR codes). QR codes are two dimensional matrix barcodes consisting of black modules arranged in a square pattern on a white background. QR codes encode information—they can store up to 2,953 bytes and are designed for fast readability.

The solution, which we call FES, Feedback Embedded System, requires users to scan QR codes displayed inside the claimed check-in venues. The QR codes change periodically and embed unique and hard to guess information that can be verified by the GSN provider to be associated with the venue. This is achieved by programming the SPOTR$_V$ equipment of venue V to generate and store a private, signing key. SPOTR$_V$ samples the time and signs it, along with an expiration interval, $T, \Delta T, S_V(T, \Delta T)$, embeds this string into a QR code and displays it onto its screen.

A user checking-in at V, is required to approach SPOTR$_V$, scan the displayed QR code, decode the embedded value and send it to the GSN provider. The provider verifies first that the value has not expired, the current time being within the interval $[T, T + \Delta T]$. It verifies the signature on the last field, using the public key associated with venue V. If either verification fails, the provider denies the check-in. Otherwise, it validates the check-in. SPOTR$_V$ changes the QR code when the expiration time $T + \Delta T$ is reached.

FES assumes the provider knows the public verification key corresponding to SPOTR$_V$'s private signing key. SPOTR$_V$ however lacks networking interfaces and the user cannot be assumed to be able to copy by hand a long and random public key. Instead, when the owner installs the SPOTR$_V$ device, she instructs it to generate a signature key pair, consisting of a public and a private key. SPOTR$_V$ stores the private key, embeds the public key into a QR code and displays it on the screen. The owner scans the QR code, decodes the public key and sends it to the GSN provider. The GSN provider associates the public key with the venue V.

The required user interaction is a drawback for FES. To check-in, users need to find the device deployed in a participating venue and scan the displayed QR code. We address this problem by taking advantage of the networking capabilities of modern devices. Many smartphones are equipped with an array of local communication media, including Wi-Fi, Bluetooth and Near Field Communication (NFC). We devise a Network Embedded System (NES) solution, where part of the check-in verification takes place at the venue: The communication between the user device and SPOTR$_V$ can take place over any local RF communication technology.

NES also assumes that SPOTR$_V$ is able to sign messages, and the GSN provider stores the public, signature verification key. In addition, NES requires each user to be able to authenticate messages. For this, during the subscription process, the user is assigned by the provider a unique, random key K. K is stored on the user's device.

The location verification protocol has two parts. In the first part, SPOTR$_V$ engages the user in a challenge/response protocol, whose small latency ensures the presence of *a* user at V. In the second part, the authenticity of the response is verified by the social network provider, ensuring that the present user is who it claims to be (and not a proxy).

Specifically, when a user checks-in at V, it sets up a connection with SPOTR$_V$, over any local RF media. SPOTR$_V$ initiates a challenge/response protocol, by sending the user the currently sampled time T, an expiration interval ΔT and a fresh random value R. The user's device generates a hashed message authentication code HMAC [45] of these values, with the key K assigned by the provider, and sends the result back to SPOTR$_V$. SPOTR$_V$ ensures that the response is received within a desired interval from the challenge. This prevents wormhole attacks, by ensuring the user has insufficient time to forward the challenge to a remote attacker. If this condition is satisfied, SPOTR$_V$ signs the received HMAC with the private key and sends the result, $T, \Delta T, R, S_V(H_K(T, \Delta T, R)$, back to the user. We use $H_K(M)$ to denote the HMAC of a message M with the key K. The last field of the message denotes SPOTR$_V$'s signature of the first fields of the message.

The user forwards the message to the provider, along with the name of the venue. The provider verifies the receipt of this packet within the validity interval $[T, T + \Delta T]$. It retrieves the key K associated with the user and the public key associated with the venue V. It then recomputes the HMAC on the first three fields of the packet with the key K and verifies SPOTR$_V$'s signature on the HMAC. If either verification fails, the provider denies the check-in. Otherwise, the check-in is validated.

FES and NES prevent a single attacker from performing fake check-ins: a single attacker not present at a venue V is unable to scan the QR codes at V or respond to the challenge presented by SPOTR$_V$. Of particular concern however are wormhole attacks. In a wormhole attack, one perpetrator located at venue V, acts as a proxy and forwards all traffic between SPOTR$_V$ and a remote attacker. This enables the remote attacker to check-in at V.

6.3 Evaluation

We have implemented our solutions in Android and Java and have tested them on (i) a Motorola Milestone smartphone featuring an ARM Cortex A8 CPU @ 600 MHz and 256 MB RAM a (ii) Dell laptop with an Intel (R) Core(TM) i7-2620M CPU @ 2.70GHz and 6,GB RAM and a (iii) 16 quadcore server with an Intel(R) Xeon(R) CPU X7350 @ 2.93 GHz and 128 GB RAM. We have used RSA to implement signatures, AES for symmetric encryption and SHA-512 for cryptographic hashes.

Wormhole Attack Resilience Wormhole attacks are best detected through timing analysis. Figure 10 shows the overhead imposed by FES and NES in the presence and absence of wormhole attacks. The overhead of FES is dominated by the cost of scanning a QR code (190 ms at 20 cm) thus hiding the additional latency of wormholes. In contrast, in NES, the low latency over local Wi-Fi communications and the small computation cost of hash based message authentication codes, make wormhole attacks stand out: the two-way wired communication between attackers imposes an overhead that is 12 times higher than the overhead of honest users. Thus, in the following, we use NES to provide location verification for the infiltration attack defenses.

Solution Overhead We have run the server side CheckIn on the three platforms mentioned above using a stream of 10,000 check-in. Figure 11 shows our results using logscale for the y axis. We have used both 1024 bit and the currently

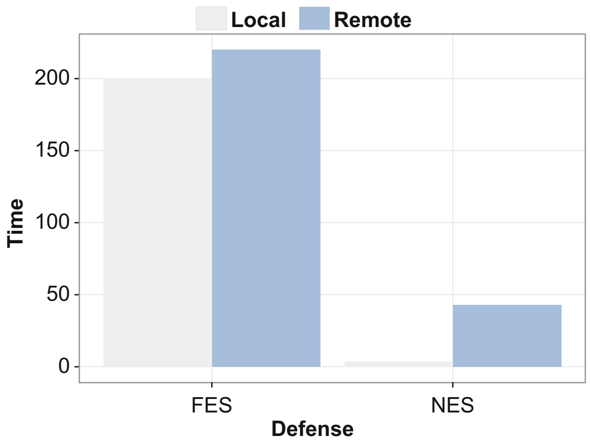


Fig. 10 Defending against wormhole attacks: FES and NES times for honest users and wormhole attackers

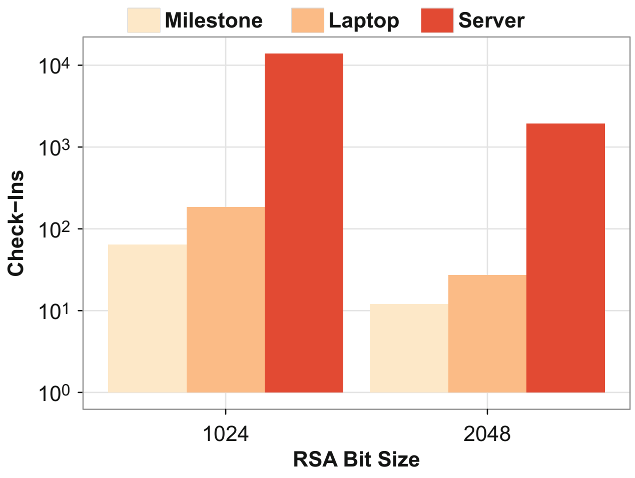


Fig. 11 Server side CheckIn performance for 1024 and 2048 RSA key bit sizes for three platforms

recommended [46] 2048 bit long RSA keys (modulus). The AES key bit size is set to 128 bits. CheckIn is efficient: The 16 quadcore server can support almost 2000 CheckIns per second, or more than 170 million CheckIns per day. Since Foursquare currently registers a few million check-ins per day [47], our results show that a server can easily support much higher loads. Even a first generation Android smartphone can support 64 CheckIns/s for 1024 bit RSA keys and 12 for 2048 bit keys.

We further study the overhead for TopoProof and TopoMatch to process check-in records during an invitation, on a smartphone. We expect the performance of TopoProof and TopoMatch to be linear with the number of processed check-in records. We tested their performance over 1000 records. The first two bars in each group shown in Fig. 12 illustrate our results. The y axis shows in logscale the time in ms taken to process these records on the Milestone. While we expected TopoProof to be slower then TopoMatch, we observe that during TopoProof, the smartphone can process almost 150 records in a second for a 2048 bit RSA modulus (200 for 1024 bit). TopoMatch requires only hash generations and can process almost 900 records per second. As our data shows, few users have more than 500 check-in records. Thus, the overhead imposed by TopoProof and TopoMatch is small, even when the verifications take place on resource constrained devices.

7 Conclusions

In this chapter we have shown that infiltration attacks are a serious social networking threat, that is not addressed by standard constructions providing data confidentiality. We have shown that location information extracted from geosocial

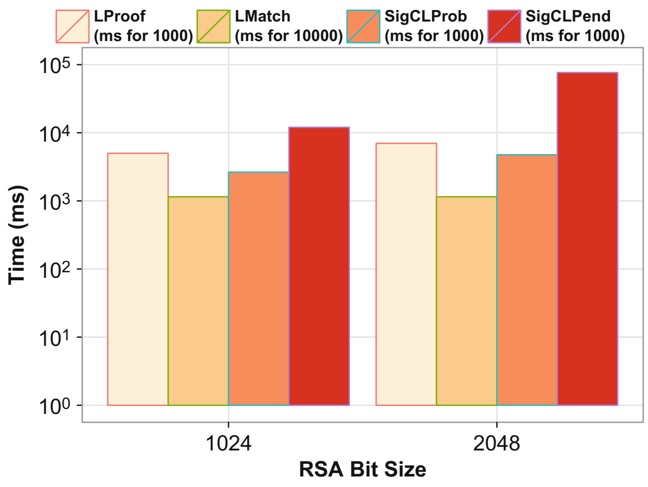


Fig. 12 Smartphone performance for TopoProof, TopoMatch and SignalCoLocation for 1024 and 2048 RSA key bit sizes

networks can play an instrumental role in detecting infiltrations, as being friend invitations received from users with whom no common context can be established. Furthermore, since location information can be easily faked in geosocial networks, we have proposed venue centric location verification solutions. The solutions are resilient to wormhole attacks.

Location information pervades most social networks today. Thus, this work applies not only to the original geosocial networks (Foursquare, Yelp) but also standard sites like Facebook or Twitter.

One obstacle to the adoption of location information as a universal verification tool, rests in the non-ubiquity of check-in information: not all users take advantage of the check-in options of their social networking sites. While this may change in time, we note that smartphones can be instrumented to periodically capture validated user locations. At a later, posting time, information may be retrieved enabling users to privately validate their earlier presence at the associated location—without revealing the precise time.

References

1. Caldwell, T.: Spear-phishing: how to spot and mitigate the menace. Comput. Fraud Secur. **2013**(1), 11–16 (2013)
2. Zinman, A., Donath, J.S.: Is britney spears spam? In: Proceedings of the Fourth Conference on Email and Anti-Spam (CEAS 2007) (2007)
3. Gartner study finds significant increase in e-mail phishing attacks: http://www.gartner.com/press_releases/asset_71087_11.html (2004, April)

4. LorneFade: Facebook Tools: A dive into facebook bots. http://www.lornefade.com/affiliate-marketing/facebook-tools-a-look-into-one-of-the-first-facebook-bots (2007, December)
5. Foursquare Hacked by TechCrunch Editor Michael Arrington (UPDATED): The New York Observer http://observer.com/2010/10/foursquare-hacked-by-techcrunch-editor-michael-arrington-updated/. (2013, Last accessed on August 10)
6. Big Boss: Location spoofer. http://goo.gl/59HMk (2011)
7. Gpscheat!: http://www.gpscheat.com/ (2015, Last accessed on June)
8. He, W., Liu, X., Ren, M.: Location cheating: A security challenge to location-based social network services. In: Proceedings of IEEE ICDCS (2011)
9. Von Ahn, L., Blum, M., Nicholas, J.H., Langford, J.: CAPTCHA: Using hard AI problems for security. In Advances in Cryptology–EUROCRYPT 2003, pp. 294–311. Springer, Berlin, Heidelberg (2003)
10. Foursquare: https://foursquare.com/ (2015, Last accessed on June)
11. Yelp: http://www.yelp.com (2015, Last accessed on June)
12. Bilge, L., Strufe, T., Balzarotti, D., Kirda, E.: All your contacts are belong to us: automated identity theft attacks on social networks. In: Proceedings of the 18th International Conference on World Wide Web (WWW) (2009)
13. Caverlee, J., Webb, S.: A large-scale study of MySpace: Observations and implications for online social networks. In: Proceedings of the AAAI (2008)
14. Nazir, A., Raza, S., Chuah, C.-N.: Unveiling facebook: a measurement study of social network based applications. In: Proceedings of IMC (2008)
15. Social Honeypots: Making Friends with a Spammer Near You. In: Proceedings of the 5th CEAS (2008)
16. Caverlee, J., Liu, L., Webb, S.: Socialtrust: tamper-resilient trust establishment in online communities. In: Proceedings of JCDL, pp. 104–114 (2008)
17. Boshmaf, Y., Muslukhov, I., Beznosov, K., Ripeanu, M.: The socialbot network: when bots socialize for fame and money. In: Proceedings of the 27th Annual Computer Security Applications Conference, pp. 93–102. ACM, New York (2011)
18. Stein, T., Chen, E., Mangla, K.: Facebook immune system. In: Proceedings of the 4th Workshop on Social Network Systems, p. 8. ACM, New York (2011)
19. Tootoonchian, A., Saroiu, S., Ganjali, Y., Wolman, A.: Lockr: Better Privacy for Social Networks. In: Proceedings of ACM CoNEXT (2009)
20. Luo, W., Xie, Q., Urs Hengartner. Facecloak: An architecture for user privacy on social networking sites. In: Proceedings of the 12th IEEE International Conference on Computational Science and Engineering (CSE), pp. 26–33 (2009)
21. Bhattacharjee, B., Baden, R., Spring, N.: Identifying close friends on the internet. In: Hotnets (2009)
22. Sun, J., Zhu, X., Fang, Y.: A privacy-preserving scheme for online social networks with efficient revocation. In: Proceedings of the 29th Conference on Information Communications, INFOCOM'10 (2010)
23. Puttaswamy, K.P.N., Zhao, B.Y.: Preserving privacy in location-based mobile social applications. In: Proceedings of the Eleventh Workshop on Mobile Computing Systems and Applications, HotMobile '10, pp. 1–6. ACM, New York (2010)
24. Sastry, N., Shankar, U., Wagner, D.: Secure verification of location claims. In: Proceedings of the 2nd ACM Workshop on Wireless Security, WiSe '03 (2003)
25. Howard, A., Siddiqi, S., Sukhatme, G.S.: An experimental study of localization using wireless ethernet. In: In 4th International Conference on Field and Service Robotics (2003)
26. Chiang, J.T., Haas, J.J., Hu, Y.-C.: Secure and precise location verification using distance bounding and simultaneous multilateration. In: Proceedings of the Second ACM WiSec (2009)
27. Zhu, Z., Cao, G.: APPLAUS: A Privacy-Preserving Location Proof Updating System for Location-based Services. In: Proceedings of IEEE INFOCOM (2011)
28. Feldman, P., Micali, S.: An optimal probabilistic protocol for synchronous byzantine agreement. SIAM J. Comput. **26**(4), 873–933 (1997)

29. Saroiu, S., Wolman, A.: Enabling New Mobile Applications with Location Proofs. In: Proceedings of HotMobile (2009)
30. Luo, W., Hengartner, U.: VeriPlace: A Privacy-Aware Location Proof Architecture. In: Proceedings of ACM SIGSPATIAL GIS (2010)
31. Egelman, S., Cranor, L.F., Hong, J.: You've been warned: an empirical study of the effectiveness of web browser phishing warnings. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1065–1074. ACM, New York (2008)
32. Wogalter, M.S., DeJoy, D., Laughery, K.R.: Warnings and Risk Communication. CRC Press, Boca Raton (2005)
33. Shin, D., Lopes, R.: An empirical study of visual security cues to prevent the sslstripping attack. In: Proceedings of the 27th Annual Computer Security Applications Conference, pp. 287–296. ACM, New York (2011)
34. Walton, D.N.: Ad Hominem Arguments. University Alabama Press, Tuscaloosa (1998)
35. Captcha buster: http://captchabuster.com (2010)
36. Viswanath, B., Mislove, A., Cha, M., Gummadi, K.P.: On the Evolution of User Interaction in Facebook. In: Proceedings of WSON (2009)
37. Watts, D.J., Strogatz, S.H.: Small world. Nature **393**, 440–442 (1998)
38. Cheng, Z., Caverlee, J., Lee, J., Sui, D.Z.: Exploring millions of footprints in location sharing services. ICWSM 2011, pp. 81–88 (2011)
39. Cha, M., Hamed H., Fabricio F., Gummadi, P.K.: Measuring user influence in twitter: The million follower fallacy. ICWSM 10 **30**, 10–17 (2010)
40. Adams, A., Angela Sasse, M.: Users are not the enemy. Commun. ACM **42**(12), 40–46 (1999)
41. Foursquare Official Blog: On foursquare, cheating, and claiming mayorships from your couch. http://goo.gl/F1Yn5 (2011)
42. iBeacon: http://en.wikipedia.org/wiki/IBeacon, (2015, Last accessed on June)
43. SensorTag: http://www.ti.com/ww/en/wireless_connectivity/sensortag/index.shtml?DCMP=lprf-stdroid&HQS=lprf-stdroid-pr (2015, Last accessed on June)
44. Hari Gottipati: With iBeacon, Apple is going to dump NFC and embrace the internet of things. http://gigaom.com/2013/09/10/with-ibeacon-apple-is-going-to-dump-on-nfc-and-embrace-the-internet-of-things/. (2013, september)
45. Mihir, B., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication, CRYPTO, pp. 1–15 (1996)
46. Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: Recommendation for Key Management Part 1: General (Revised). NIST Special Publication 800-57 (2007, March)
47. Lauren Indvik: Foursquare Surpasses 3 Million User Registrations. http://mashable.com/2010/08/29/foursquare-3-million-users/, (2010, August)

An Economical, Deployable and Secure Architecture for the Initial Deployment Stage of Vehicular Ad-Hoc Network

Baber Aslam, Ping Wang, and Cliff C. Zou

Abstract With the fast development of vehicular ad-hoc network (VANET) and related technologies in both academia and industry, many VANET systems have been presented in recent years. However, the majority of them have the assumption that all or most vehicles have installed with wireless communication devices and an elaborate roadside infrastructure exists. This assumption is not realistic for the critical and long transition period of VANET, when only a small portion of vehicles will be equipped with wireless devices (we refer to them as smart vehicles) and limited roadside infrastructure will exist. In this chapter, we present an economical, deployable and secure VANET system design that could facilitate the gradual deployment of wireless communication among vehicles. The system design is intended to stimulate VANET adoption without the need of elaborate infrastructure, large number of smart vehicles, huge investments by service providers or expensive end user devices. Economical Roadside Units (RSUs) that do not need expensive Internet access (especially in rural areas) can be incrementally deployed along critical road sections. They behave as temporary (traffic) information storage and relay points to serve any passing-by smart vehicles, while smart vehicles report/receive information to/from RSUs and relay information between RSUs. In addition, we present a public-key infrastructure based security architectures centered on these RSUs. We show that we can achieve connectivity with a high degree of confidence using only a small number of smart vehicles and RSUs. We present workable models for multi-confidence level data verification and time-location based secure positioning systems, along with possible threats and their defenses within the scope of our proposed designs.

B. Aslam
National University of Sciences & Technology, Islamabad, Pakistan
e-mail: baber-mcs@nust.edu.pk

P. Wang
Symantec Corporation, 1001 Heathrow Park Lane, Heathrow, FL 32746, USA
e-mail: jenpwang@gmail.com

C.C. Zou (✉)
Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA
e-mail: czou@cs.ucf.edu

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_16

1 Introduction

In vehicular ad hoc network (VANET), the networking architecture could be infrastructure-based, or ad hoc based, or most likely, combined together. Vehicle to Vehicle (V2V) communication is ad hoc and Vehicle to Infrastructure (V2I) communication is infrastructure-based through access points (i.e., roadside units). The roadside units (base stations) are then connected to the Internet and provide necessary services to vehicles. The provision of services largely depends on connectivity of the roadside units to each other and to the Internet. The success of VANET depends on the existence of mature roadside infrastructure and sufficient number of vehicles equipped with wireless communication devices (we refer them as "smart vehicles"). Most VANET researches are based on either or both of these requirements.

However, both of these requirements will not be realistic during the initial years of VANET deployment. It will not be economically feasible to initially install a large number of fully networked roadside units to cover a region. Further, during the long transition period, in most cases, there will not be sufficient number of smart vehicles to enable comprehensive V2V communication, which is an essential element in many VANET applications [1]. The roadside infrastructure will remain uneconomical in rural areas even after the initial deployment since, in many rural areas, there will not be sufficient number of smart vehicles for years to come. V2V communication between vehicles traveling in opposite direction is very important for effective routing of messages [2]. This communication may not be possible in some locations due to road layout (such as when two directions of a road are far away from each other) or because of uneven distribution of traffic on opposite driving directions (such as high outbound/low inbound traffic from/to residential areas during morning hours).

Although there is plenty of research on VANET, the solutions to the challenges existing during the long transition period in VANET deployment, as discussed above, are largely ignored. There are some research solutions that offset the absence of roadside architecture by either not relying on it (i.e., using only V2V communication), or using cellular architecture, or using existing available Wi-Fi hotspots, or using static/mobile relay units (Delay tolerant networks) [3–8]. Most of these approaches target specific applications and may not be easily upgradable (during later stages) to VANET architecture (such as those defined by IEEE P1609 working group). There is also lack of research on how to provide incentive for vehicle owners to install wireless devices when support from roadside infrastructure is insufficient and very limited number of smart vehicles are on the road.

In this chapter, we present an economical, deployable and secure VANET system design to solve these challenges. Our focus is to provide a transitional/interim solution that can be used to start up (or give impetus) to VANET activities during the long initial transition period by making VANET easy to deploy, secure and economical. At the same time the design should be incremental/progressive and should be easily transformed into architectures that are already specified in VANET standards/protocols without requiring any major revamp/modifications. From now on we will mostly refer "smart vehicles" as "vehicles" without considering any vehicle that has no communication device, unless there is a need to explicitly mention smart vehicles.

The contributions of this chapter are manifold. First, we present a VANET system design that is economical, realistic, incremental and deployable during the initial long transition period when smart vehicles have a low penetration rate. The proposed design also ensures V2V communication in the above mentioned situations which are necessary for the success of VANET. Core component in our design, the Roadside Units (RSUs), can be standalone with minimum intelligence in their basic form. Our proposed design does not require RSUs to be interconnected or connected to the Internet. We present a basic protocol that makes the communication between roadside units possible via mobile vehicles. The simulation results indicate considerable performance gains by just using standalone RSUs.

Second, the simplified security architecture proposed, where only RSUs have certificates, enables an economical and strong information assurance in VANET in the similar way as the public-key infrastructure used in Internet web service—RSUs behave like web servers with certificates and vehicles behave like client web browsers. In this way, the mature Internet public-key infrastructure can be directly deployed in VANET without the complicated and expensive requirement of digital certificate in every dynamically-located vehicle.

Last, although the proposed design is simple and inexpensive, it can be effectively used to provide advance security features such as data verification and secure positioning. We present multi-confidence level data verification and time-location based secure positioning systems based on the proposed system design. We also analyze possible threats and the protection offered by our design against these threats.

The chapter is organized as follows. In Sect. 2 we present related work. Section 3 describes the challenges. Section 4 gives detailed description of our proposed design. Section 5 highlights the security features of our design. Section 6 analyzes possible threats and their defenses. Section 7 presents the simulation details. Section 8 presents discussion and Sect. 9 gives the conclusion.

2 Related Works

Most of the existing research in VANET presents routing algorithms for V2V communication [9–11]; these protocols rely on the assumption that sufficient number of vehicles will be available for relaying messages. Some of the research work also addresses the routing in disconnected or intermittently connected networks [12–15]. A hybrid approach has also been presented to address limited connection time between vehicles [16]. However, during the initial deployment there will not be sufficient number of smart vehicles on the road to even form small clusters for these protocols to work. Further, lack of roadside infrastructure will also make use of hybrid protocols difficult.

The technique for transmission of data between nodes of a disconnected or partitioned network using temporary storage at intermediate nodes is a delay tolerant network (DTN) [3]. Besides satellite networks, the concept of DTN has been widely applied to VANET (which may be considered as DTN especially during initial deployment stages) [3–5, 12, 17, 18]. It is pertinent to highlight here that most of the existing research in this context is on V2V communication protocols [4, 12, 17] where mobile nodes temporarily store the message if no route is available and later opportunistically forward the message. These protocols may be used to solve the disconnected network problem due to uneven distribution of traffic, but may not be an effective solution to low penetration rate issues. Throwboxes in UMass' DieselNet [5] are similar to our standalone RSUs. The throwboxes act as stationary routers to improve connectivity among mobile nodes (buses) that are equipped with multiple radios (including a long-range radio), GPS recording devices etc. In our research the RSUs are not just the routers, but in addition they also receive, process and disseminate information (such as safety or warning). Mobile nodes in MIT's CarTel [18] are equipped with multiple sensors; the data collected from these sensors is processed and transferred to a central portal by these nodes. The transfer is accomplished opportunistically via Wi-Fi (hotspots, roadside units), Bluetooth or by nodes themselves (data mules). A specially designed delay-tolerant network stack (CafNet) is used for communication. Our emphasis, in this chapter, is not on making major modification to existing VANET standards/protocols, but to enable their gradual/incremental deployment during initial phases.

Infostations architecture allows use of high speed and generally dispersed access points. The access points/stations afford transfer of high volume of data at cost of connectivity. They can be especially useful in VANET environment where vehicles are moving at fast speeds and connection time to access points is very limited [19, 20]. This architecture cannot solve the low penetration problem since the infostations will generally be widely dispersed. Further, these must also be fully networked with Internet, which will be expensive to install and maintain.

A number of researches have incorporated cellular networks in VANET [6, 7, 21, 22]. Cellular networks are mostly used as a backbone—a replacement to roadside infrastructure. Cellular networks, though pervasive, offer lower data rates as compared to Wi-Fi (roadside infrastructure). Although with the advent of 3G/4G technologies data rates close to broadband can be achieved, these technologies are not uniformly available throughout cellular coverage areas and many users are still dependent on other heterogeneous technologies (WAP, GPRS, EDGE, HSDPA, etc—[23]). Further, cellular data plans subscriptions are expensive; an unconstrained plan with a 5GB/month limit costs approximately $700/year. 5GB/month means per day a user on average can send/receive 50 emails (20 with attachments), download a song and a game/app, view 40 web pages, posts 10 social media posts with photos, and watch a streaming video of 40 min [24]. Although unlimited data plans and those that cost few ten of dollars are also available, but these have several fine print conditions; such as 'usage patterns' (no file sharing, excessive usage, etc), 'can only be used on smart phones' (no tethering), 'can only access certain service' (email, predefined websites, etc), and ‘must have a qualifying voice plan’. Some service providers are charging approximately $2/MB or 1¢/KB for web browsing. All major cellular service providers are now offering and encouraging users (such as offering ‘unlimited Wi-Fi usage with data plan’) to access data through hotspots (which use Wi-Fi just like VANET roadside infrastructure instead of 3G/4G). This also highlights cost/benefit of Wi-Fi over 3G/4G. In addition cellular networks also have few other disadvantages such as expensive to built/maintain, billing/licensing issues among different service providers, higher roaming rates, large and variable latency, central switching/resource management, difficult to scale and occasional blackouts [8, 23, 25–29].

A class of protocols uses the store-and-forward approach for V2V communications [9, 30]. MDDV [9] is a multi-hop V2V protocol. It uses predictability of vehicle movement to route the messages. It assumes the vehicles to be equipped with GPS and digital maps. It uses trajectory and geographical based forwarding. If end-to-end path does not exist then messages are stored and later forwarded when a connection is established. VADD [30] is also a multi-hop protocol using the carry-and-forward paradigm. In this protocol a vehicle carries a message until it finds another vehicle in communication range, then it forwards the message. It assumes that the vehicles are equipped with GPS, digital maps and also have detailed traffic statistics such as vehicle density, vehicle speeds. It bases its decision of message-forwarding on these statistics. Both the protocols [9, 30] are used to transfer messages between vehicles in multi hops.

Lochert et al. [31] compare the performance of standalone and networked stationary supporting units (SSU) in context of low penetration rate. The work focuses on dissemination of information from a central point in a city scenario. They show that the networked SSUs (connected via a backbone) improve the performance dramatically as opposed to the standalone SSUs. V2V communication also plays an important part in their scenario. Whereas in our case we used very limited penetration rate so that V2V communication is not possible and our results show that standalone RSUs do increase the performance.

Our work comes closer to protocols that use vehicles to transfer messages between roadside units [32–34]. Chuah and Fu [35] present a protocol using multi-hop V2V communication between roadside units. They present a detailed mechanism for forwarding of messages at each hop. It makes use of query and response messages at each hop. Petit et al. [32] present a set of protocols for data relaying between roadside units using vehicles. The protocols give different options for transfer of data between a source/sink and a vehicle. It uses solicit and beacons for selection of appropriate vehicle to carry the data. The work has been further extended by Mansey et al. [34] giving vehicle-roadside unit data transfer mechanisms and reliable multi-packet data transfer schemes. The protocol does not provide details on routing between different roadside units. Ding et al. [33] present a static node assisted adaptive routing protocol. It is basically a multi hop protocol that makes use of static nodes at the intersection to store and forward the messages, thus improving performance over other multi hop V2V communication protocols.

The majority of security solutions for VANET are based on public key infrastructures, where in addition to roadside units, every vehicle is also assumed to have a certificate [36–42]. Many temporary certificates (pseudonyms), instead of one permanent certificate, are usually employed to ensure privacy [36, 37]. These pseudonyms are either stored in bulk (in a tamper proof device—TPD—[36]), or issued by an online authority [37, 38], or generated by the user himself [39, 40]. The centralized certification authority (CA) based solutions present a number of challenges which may be difficult to address during the initial deployment stages of VANET. The CAs must be organized in a hierarchical manner for effective management and a trust relationship must exist among regional CAs. This means certificate verification may take longer especially if the trust relationship goes through a long chain. Further, it also makes revocation difficult since revocation list (RL) must be distributed to all regions as vehicles are not restricted to remain within their regions. The RL may grow over time, making its distribution more difficult. Papadimitratos et al suggested restricting the scope of RL within a region, requiring visiting nodes from other regions to obtain temporary certificates [43]. In current designs, too much trust is placed on TPD, which stores all cryptographic materials (permanent certificate and pseudonyms), performs cryptographic operations (signing/verifying messages) and processes revocation messages/commands (erase keys/pseudonyms when revoked) [36]. Since the vehicle (and TPD) cannot be physically guarded as other electronic security devices (smart cards etc), those requirements will make the device quite expensive [44]. Further, the pseudonyms, when exhausted, must be reloaded thus requiring a periodic maintenance.

Our research work differs from above mentioned protocols in many ways. We do not assume vehicles to be equipped with GPS and digital map, or have road statistical data, which makes our design more realistic especially in the initial transition period. Our design does not involve V2V communication, thus it works well when smart vehicles are sparsely distributed on roads. We do not assume roadside units to be always connected to infrastructure (i.e., fully networked or connected to the Internet), which makes the RSU deployment in our design economical and practical during the transition stage. We present an integrated design involving vehicles and roadside units with varying degree of capabilities. Besides being economical, the design is also scalable and can easily be upgraded without any major modifications in protocol. In addition our design offers security without much complexity; we do not require smart vehicle to have certificates which simplifies certificates issuance and revocation.

3 Challenges in VANET Initial Transition Stage

The success of VANET depends on efficient functioning of two major communication components: V2V and V2I communications. The challenges that will be faced during the initial deployment phase and, in some cases, later phases as well, are listed below:

- Insufficient number of smart vehicles to enable V2V communication or to provide any useful information (such as accidents reports) [14]. Reduced connection time among vehicles due to low smart vehicle density [16].
- Disconnected V2V networks during low traffic hours even with high network penetration [13, 14].
- Lack of elaborate roadside infrastructure—from economical considerations the installation of roadside units will take a long time and will be incremental. This will still be the reality in rural areas even after the end of initial deployment phase.
- Lack of V2V communication between vehicles traveling in opposite direction either due to large distances between opposing lanes or because of vehicles reaching the point of contact at different times.
- Very small V2V communication window between vehicles traveling in opposite direction.
- Difficult and expensive vehicle-based certificate management and operations (such as initiation, maintenance and revocation of certificates to all vehicles) during the initial deployment due to non-availability of pervasive connectivity to certification authorities and the dynamic locations of each vehicle. Further, it may be economically unattractive for a vehicle owner to pay for issuance and renewal of his vehicle's certificates.
- Difficult to obtain secure positioning information due to limited roadside infrastructure and smart vehicles.
- Difficult data verification due to the sparse density of smart vehicles for the process of data correlation. Further, it is not practicable for each vehicle to conduct its own data verification.

Our proposed design addresses all of the above challenges in a simple, efficient and economical way. We propose use of roadside units that are very simple and economical to facilitate VANET communication. We also propose two different security architectures that require minimal investment from both service providers and users.

4 Proposed Design

The common characteristic of all VANET applications is either collection or dissemination of information from/to vehicles in a timely and efficient manner. V2V and V2I communications complement each other in achieving this flow of information. For example, we can overcome the issue of low market penetration of smart vehicles by having more elaborate roadside infrastructure (i.e., passing information through fully networked roadside units or using infrastructure to infrastructure—I2I communication), or conversely, high penetration can overcome lack of roadside infrastructure or I2I communication (i.e., passing information using V2V communication). As discussed earlier, during the initial stage of VANET, both

V2I (also I2I) and V2V communications will not be very effective. So we need to address the issues of V2V and V2I connectivity in an efficient and economical way. Since we cannot influence the market penetration of smart vehicles, the other solution is to improve V2I and I2I communications; which will, in turn, complement the lack of V2V communication. One option is to have pervasive fully networked roadside infrastructure (to improve V2I and I2I communications). Though it may be possible to have such a network in urban-areas but in rural-areas/along-highways (where there is not much manmade infrastructure) this option will be quite expensive and impractical. Another option is to use cellular network as a replacement to roadside infrastructure. Though cellular networks are pervasive, but in addition to the technical and economical disadvantages mentioned earlier in Sect. 2, this option will also introduce heterogeneous technologies (i.e., typical VANET architecture in urban-areas and cellular based architecture in rural-areas/along-highways); making the transition to final VANET architecture difficult.

We suggest improving connectivity/communication by using roadside units (RSUs). In its basic form our proposed RSU is standalone with only store-and-forward capability, which makes it economical and easy to install/maintain, especially in rural-areas or along highways. Other types of RSUs include those that are locally connected to each other or connected to the Internet (those located close to manmade infrastructure or in urban-areas). Details of RSU design are given in Sect. 4.1. An overview of RSU's role in achieving different connectivity requirements is given below:

- **V2V Communication** Direct multi-hop V2V communication will not be possible during initial deployment phases due to low market penetration. V2V communication among vehicles with spatial displacement will be improved using store-and-forward capability of RSUs and those with temporal displacement will be improved if RSUs are networked. Besides broadcast communication, one-to-one communication among temporally displaced vehicles may also be achieved with the support from RSUs if vehicles have fixed routes and travel schedules (as in the case of daily commute).
- **V2I Communication** Economical and easy installation/maintenance features of RSUs help in achieving high RSU densities even in rural areas and along highways; this will help in improving V2I communication.
- **I2I Communication** V2I communication is of little use if there is no I2I communication. RSUs that are connected to each other or to the Internet can easily communicate with each other. Standalone RSUs can use passing-by vehicles as relays to communicate with each other.

Our design integrates RSUs of varying capabilities thus making architecture economical, easy to install/maintain, incremental/progressive (basic RSUs can easily be upgraded to higher capability ones), homogenous (same technology in urban and rural areas) and upgradable to final VANET architecture.

4.1 Roadside Unit: RSU

The motivation of our design is to make roadside units light weight, simple/easy to install and economical. Our proposed design does not require all RSUs to have the same capabilities. Multiple versions of RSUs enable engineers to have necessary flexibility in designing a VANET architecture that is suitable to their requirements and budget.

4.1.1 Multiple Versions

We define several different versions of RSUs with varying capabilities/functions and network connectivity. In its basic version, an RSU is a standalone unit with temporary store-and-forward capability. In terms of connectivity, RSUs can be standalone, locally networked (via wire or wireless such as WiMax—[45–48]), connected to the Internet via wire or wireless, or just have backend receiver-only capability in order to receive data from satellite, cellular, commercial radio, etc. RSUs may have sensors for monitoring local weather, road condition, traffic, etc. All RSUs are tamper proof, capable of receiving and sending data from/to vehicles and have some information processing capabilities. Possible versions of RSUs are listed in Table 1. A possible architecture with standalone, locally connected and globally connected RSUs is shown in Fig. 1.

Store-and-forward is the basic capability and enables an RSU to transfer messages between spatially and temporally displaced vehicles. Intelligent information processing gives RSUs the capability of encryption/decryption, data verification, provision of time/location stamp, certificate revocation, etc. Sensors are used to collect local traffic and weather data. This collected data can be used for verification of data provided by vehicles. Limited local connectivity means an RSU is connected

Table 1 Different version of RSUs with increasing functionality. An RSU with a larger version number will be more expensive but provide more functionality

Version	Store and forward-repeater	Intelligent with information processing	Sensors to collect local data–traffic, met etc	Limited local connectivity	Backend receive only–radio, satellite	Backend duplex connectivity
1.0	Yes	No	No	No	No	No
1.1	Yes	No	No	Yes	No	No
1.2	Yes	No	No	Yes	Yes	No
2.0	Yes	Yes	No	No	No	No
2.1	Yes	Yes	No	Yes	No	No
2.2	Yes	Yes	Yes	Yes	Yes	No
3.0	Yes	Yes	Yes	Yes	Yes	No
3.1	Yes	Yes	Yes	Yes	–	Yes

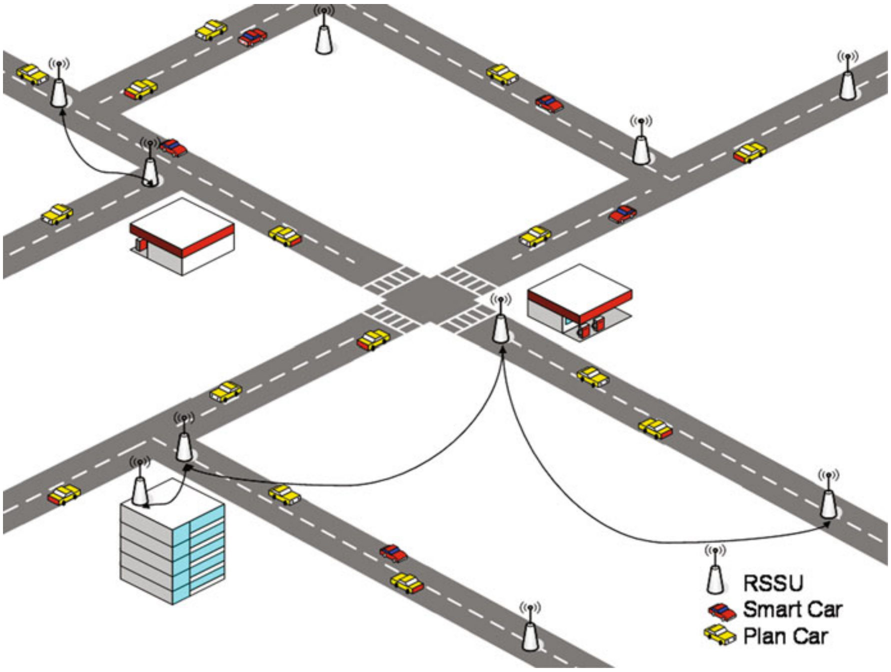


Fig. 1 The proposed architecture consists of RSUs deployed along the roads. RSUs can be standalone (the three RSUs on *top right*), locally connected to adjacent RSU (two on the *up-left corner*), or connected to Internet infrastructure (three on the *bottom*). What versions of RSUs to install depends on overall budget and services we want to provide [49]

to at least one adjacent RSU. "Backend receive only" enables reception of critical safety information, certification revocation lists, etc. It is an economical way to receive important non-local messages for dissemination to vehicles in an area, such as fire, flood, and earthquake emergency warnings. It can also be used for distribution of certificate revocation lists to RSUs similar to [36]. "Backend duplex connectivity" means connection to the Internet; such RSU can send and receive data to/from the Internet. Other RSUs can connect to the Internet through the backend duplex connected RSUs.

4.1.2 Deployment

Different versions of RSUs help in achieving economical deployment across diverse areas. In urban areas where Internet connectivity is pervasive, it is economical and easy to deploy Internet connected RSUs, whereas in rural areas or along highways where it is difficult/expensive to extend Internet connectivity (though a limited number of RSUs may be connected to Internet through cellular network), it is more economical to deploy standalone or locally networked RSUs. These basic standalone RSUs will help in getting VANET started and later they may be replaced with more advanced ones without any system overhaul.

Each RSU will have a distinct identification and an associated digital signature certificate. The certificate issuing authorities for RSU's certificates may be organized on area/region basis. All vehicles in VANET will have certificate-issuing-authority's certificate and will be capable of verifying RSUs' certificates and messages signed by these certificates. At startup, a VANET can even have Version 1.0 RSUs without any certificates. These RSUs can be used for only store-and-forward functions and be deployed at non-critical locations.

Each RSU will be aware of local map, its own location and locations of other RSUs in the area. Additionally, each RSU will also maintain a routing table with known path to each of the other RSUs in the area. Initially, this information will be added at the time of installation and later it will be updated periodically via RSU update messages. For this purpose, each RSU will periodically exchange signed Hello messages (containing routing table etc) with its neighbors. Routing-table-update procedures from any existing table-driven routing protocol may be employed for routing table updates and the details are hence omitted here. Routing between standalone RSUs relies on the relay by passing-by vehicles and is directly related to traffic density. If traffic density varies considerably during different times of day then the routing table may contain multiple entries accordingly (e.g., one each for morning commutes, one for evening commutes, and one for rest of the day).

4.2 V2V Communication

V2V communication is an important part of VANET; many VANET applications (such as cooperative driving, and safety warnings) depend on V2V communications. However, there will be a very small number of smart vehicles during the initial deployment phase of VANET and V2V communication will hardly exist. In such case standalone RSUs (with store-and-forward capability) can play an important role in achieving limited V2V communication. A sending vehicle sends a message to a nearby RSU, which stores and later forwards it to another passing-by recipient vehicle. Though this type of communication cannot be used for time-critical messages but it can still serve as a means to broadcast non-time-critical messages. If a vehicle has its certificate then it may also sign the message to ensure its authenticity to a receiver. In this way, a malicious vehicle transferring fake messages will be held accountable.

If an RSU is networked (locally or with Internet), then the RSU can support V2V communication between spatially displaced vehicles. This could be useful in quick dissemination of information within the network or across networks (if the RSU is connected to the Internet).

4.3 V2I (Vehicle to RSU) Communication

Our proposed design enables service providers to deploy a relatively large number of RSUs with less investment thus enabling more V2I communications. Each RSU will advertise its existence and offered services by broadcasting periodical beacons. The services offered by a particular RSU will depend on its version/capabilities, e.g., an Internet connected RSU may offer email service whereas a standalone RSU may only offer store-and-forward service. If an RSU is capable of sensing nearby vehicles, it can broadcast its beacon only when a vehicle enters its broadcast range—this conserves power in low traffic conditions. The beacon broadcasting interval (BI) can be defined by the maximum allowed driving speed (s) and broadcast zone diameter (Z_d), i.e., $BI = Z_d/s$.

The beacon will include an RSU's ID, certificate, location, current time, location of adjacent RSUs, services offered and critical safety information. Critical safety information is included in beacon to reduce the information relaying time. The beacon message will be signed by its issuing RSU. Critical safety information messages may also be broadcasted independent of the beacons. In this case, critical safety messages will be given priority over other messages. They will be signed by sending RSUs and will include location of sending RSUs and current time. Vehicles may relay these messages to other passing-by vehicles.

4.4 I2I (RSU to RSU) Communication

I2I communication plays a vital role in both V2V and V2I communications. I2I communication may be considered a part of V2I communication especially when roadside infrastructure is fully networked. We consider I2I communication separately because in the initial deployment stage of VANET RSUs are not necessarily connected to each other or to the Internet. Data transmission will normally be limited to adjacent RSUs only. However, there may be situations when a message is needed to be sent to another RSU that is many hops away, such as sending information about a malicious vehicle to an RSU that is known to be connected to the Internet, or relaying an accident report to emergency vehicle that is known to be located near a particular RSU.

I2I communication, depending on the connectivity of RSUs, can be divided into two types, i.e., I2I Direct communication and I2I Indirect communication, which will be introduced next.

4.4.1 I2I (RSU to RSU) Direct Communication

Some RSUs may be locally connected to adjacent ones. This connectivity can be wired or wireless. Local connectivity is economical as compared to global connectivity (to the Internet). If two RSUs are connected to each other then direct I2I communication will be used. For this, existing protocols (such as those defined by

IEEE P1609 working group) may be used, and the details are hence omitted. There may be a case where part of networking route is connected and part is disconnected; the connected part will use direct communication whereas the disconnected part will use indirect communication introduced in the following.

4.4.2 RSU to RSU Indirect Communication (via Vehicles)

If RSUs are not locally connected, then the RSUs communicate with each other using passing-by vehicles. A reputation system may be used to solicit vehicles' cooperation in relaying these messages. The exact details of such a reputation system are out of the scope and are not discussed in this chapter. The addressing information will include the destination RSU's ID and its location. If the message is not for adjacent RSUs but several hops away, routing information will also be included. Routing information will include locations and IDs of all intermediate RSUs along the path to the destination RSU. The message will be signed by its originator and any confidential information will be encrypted. The certificate of the originator will also be appended with the message.

The basic idea of *opportunistic routing* is used in our design. In opportunistic routing as opposed to deterministic routing, the node that forwards a message is not predetermined. It is determined on the fly, normally by a subset of nodes that receive the broadcast [50, 51]. An RSU broadcasts a message to every vehicle in range. There are two possible options for the selection of relaying vehicle. In the first option, after receiving the message, each vehicle waits for a random amount of time and then acknowledges the message. On hearing the acknowledgement sent by one vehicle all other vehicles will discard the message. Therefore, only one vehicle that acknowledges first is selected as the message-relay-vehicle.

One possible problem can occur for this option when the relaying vehicle diverts from the route before delivering the message. In this case the probability of success can be increased by letting more than one vehicle to relay the message. Another possible issue is the hidden-node problem (note that the small number of smart vehicles during the initial stage of VANET deployment will reduce the chances of having a hidden-node); in this case more than one vehicle will acknowledge and carry the message. This operation will provide redundancy to the protocol, but at the same time, it will require duplicate suppression at the destination. (Mathematical analysis of the number of nodes required to deliver a message with certain probability of confidence is discussed later).

Acknowledgement messages will be restricted to only one hop. End-to-end acknowledgement may be included as an optional service. The calculation of acknowledgement timeout is discussed later.

4.4.2.1 Operation

When an RSU broadcasts a message, each receiving vehicle compares the destination location with its direction of travel and discards the message if it is for an RSU on the opposite direction. For a vehicle not equipped with GPS, we have two options to determine its direction of travel relative to the location of destination RSU. First, the vehicle can use the location of RSU it has just passed and the location of current RSU to determine its direction of travel. Second, in the message the sending-RSU can include the previous-RSU's ID that a vehicle must have passed, if it is along the desired direction.

A relaying vehicle passes the message to each intermediate RSU that is listed in the routing path of the message. When an RSU receives a message, it checks message integrity and then sends an acknowledgement to its immediate upstream RSU according to the routing information contained in the message. If the message has been received before, it is discarded and only the acknowledgement is sent. This ensures duplicate elimination on a per hop basis.

If the message receiving RSU is not the destination RSU, it rebroadcasts the message to the next RSU in the routing path. It then waits for an acknowledgment from the next RSU; waiting time is defined by acknowledgement-wait-time (details in next section). If no acknowledgement is received till the expiration of acknowledgement-wait-time, it rebroadcasts the message. The process is then repeated for a fixed number of times. This guards against network overloading since there may be the cases when a message has been received but acknowledgement cannot be sent due to lack of upstream vehicular traffic. The acknowledgement generated by the destination RSU may be sent back to the source RSU as an optional service. An example flow of message and its acknowledgements is shown in Figs. 2 and 3.

A vehicle may deliver the same message to more than one consecutive RSUs, for example, in Figs. 2 and 3, vehicle V_2 delivers message M_1 to RSU_2 and

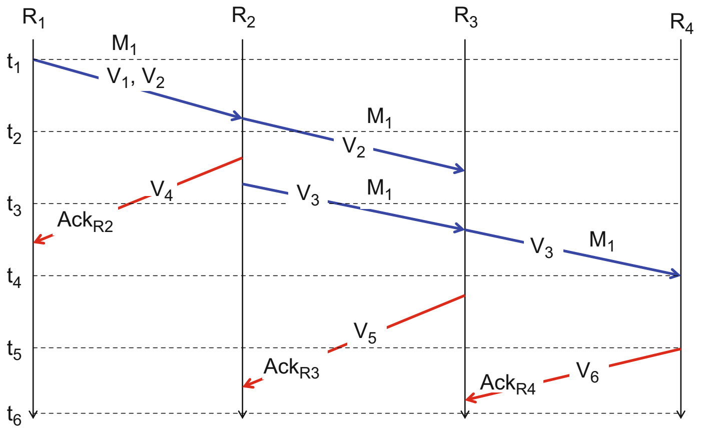


Fig. 2 Flow of a message and its acknowledgement between RSU_1 and RSU_4 (illustrated also in Fig. 3). Message M_1 is sent from RSU_1 to RSU_4 via RSU_2 and RSU_3. t_i represents time in order. R_i represents RSU_i. V_i represents vehicle i that carries a message. Ack_{Ri} is the acknowledgement message from RSU_i to RSU_{i-1}

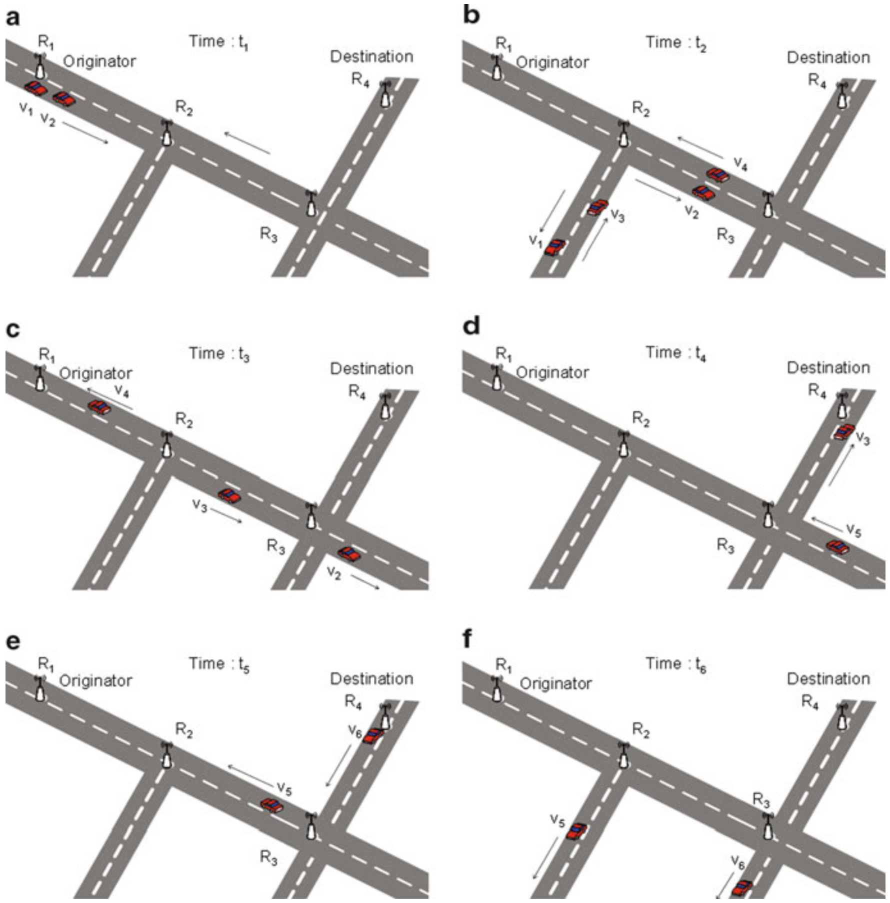


Fig. 3 Snapshots of the road conditions when a message is sent from RSU_1 to RSU_4 via RSU_2 and RSU_3 (illustrated also in Fig. 2). (**a**) V_1 and V_2 receive the message from RSU_1. (**b**) V_1 and V_2 deliver the message to RSU_2, V_1 diverts to its right at road junction, V_3 and V_4 approach RSU_2. (**c**) V_2 delivers the message to RSU_3, V_3 receives the message from RSU_2, V_4 carries the acknowledgement message from RSU_2 for RSU_1. (**d**) V_3 delivers the message to RSU_4, V_5 approaches RSU_3. (**e**) V_6 receives the acknowledgement message from RSU_4 for RSU_3; V_5 carries the acknowledgement message from RSU_3 for RSU_2. (**f**) The acknowledgement messages delivered by V_6 and V_5 to RSU_3 and RSU_2 respectively [49]

RSU_3. In order to take advantage of this situation, each receiving RSU waits for acknowledgement from its next RSU on the routing path before re-broadcasting the message, since the vehicle that has delivered the message may also deliver the message to the next RSU. But if the traffic density is low, the receiving RSU may rebroadcast the message before the end of the wait timer (to simplify the logic the RSU may rebroadcast the message before starting the wait timer).

Acknowledgement Wait Time

Each RSU waits for its acknowledgement before retransmitting. The wait time (W_t) depends on the distance to the next RSU, the average speed of vehicles and traffic conditions. It is directly related to distance (L) and inversely related to vehicle speed (s) and traffic density (d) (upstream),

$$W_t = 2\frac{L}{s} + \frac{1}{ds} + \varepsilon \tag{1}$$

where ε is a constant which caters for processing done at a node before sending the acknowledgement.

The final wait time will be estimated using Eq. (3). Here α is the smoothing factor, M is the acknowledgement arrival time and D is the smoothed deviation (similar to TCP round-trip-time estimation model—[52])

$$D = \alpha D + (1 - \alpha)\,|W_t - M| \tag{2}$$

$$\text{TimeOut} = W_t + 4 \times D \tag{3}$$

Number of Relay Vehicles

We cannot be sure that a vehicle which has passed the source RSU and is carrying the message will always pass the destination RSU without diverting on the way. Therefore, we want to estimate the number of times a source should relay the message to have some degree of confidence that the message will reach the destination.

Suppose between two RSUs, there are one or several road diversions. Among the traffic flow entering from the source RSU, only p fraction of flow goes to the destination RSU. N represents the number of vehicles passing the source RSU; the random variable X represents the number of vehicles that have passed by the source also pass the destination RSU (Fig. 4). Let's find out how many vehicles (N) should

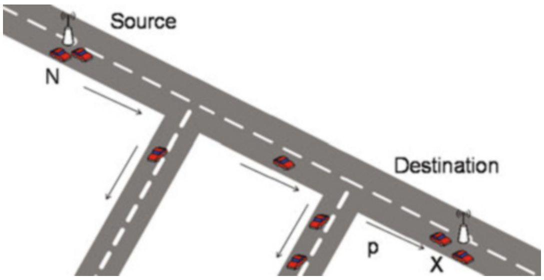


Fig. 4 Number of relay vehicles depends on the probability of vehicles passing the destination. N is the total number of vehicles passing the source, X is the random variable representing the number of vehicles that have passed source will also pass the destination

the source RSU ask to carry the message, in order to let the destination RSU have at least k vehicles passing through it, with a confidence of probability Pc (such as 95 %).

Because each vehicle has an independent probability p to go to the destination RSU, the random variable X follows *Binomial distribution*. If we denote $f(n; N, p)$ as the probability of exactly n vehicles going through the destination, then according to Binomial distribution, we can derive:

$$f(n; N, p) = \binom{N}{n} p^n (1-p)^{N-n} \tag{4}$$

The question we asked above means that the probability of having less than k vehicles passing through the destination RSU must be no more than $1 - P_c$. Thus the following inequality formula must be satisfied:

$$f(0; N, p) + f(1; N, p) + \cdots + f(k-1; N, p) \leq 1 - P_c \tag{5}$$

For $k > 1$ (which will be the case if we want more than one vehicle to deliver the message for redundancy or security purposes) formula (5) does not have a closed-form solution. To derive the value of N, we can test $N = 1, N = 2, N = 3, \ldots,$ until we find the smallest value of N satisfying the formula.When $k = 1$, the above formula means that the value of N must satisfy:

$$(1-p)^N \leq 1 - P_c \tag{6}$$

or

$$N \geq \frac{\log(1 - P_c)}{\log(1-p)} \tag{7}$$

Formula (7) gives the minimum number of vehicles that required to carry a message in order for at least one vehicle passing the destination with certain confidence level. Figure 5 shows the number of vehicles required for different values of p.

Protocol Simplification Based on GPS Data

The large scale use of GPS technology has made GPS devices economical; it is likely that in the near future all modern vehicles will be equipped GPS devices, which provide valuable data such as up-to-date location, direction and speed. In addition, when a GPS device is used for navigation, it can provide destination and trajectory information. The additional data can be used to make the communication protocol simpler and more efficient.

An RSU can query a passing-by vehicle for destination and trajectory information. Based on this information, the RSU can decide whether or not to choose the vehicle for forwarding messages to other RSUs. This will reduce the number

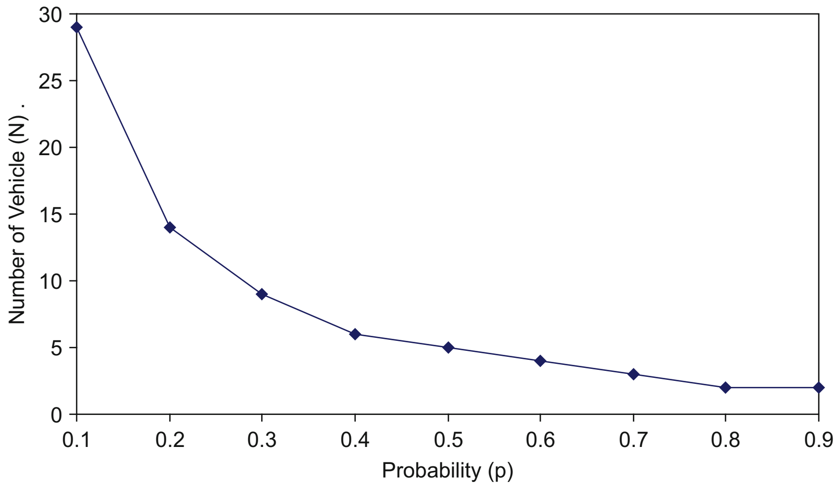


Fig. 5 Number of relay vehicles (*N*) required to deliver a message to the destination (by at least one vehicle) with a 95 % probability of confidence (*Pc*) for different probabilities (*p*) that a vehicle passing the source will also pass the destination. For example, if $p = 0.5$ and $Pc = 0.95$, we get $N = 5$ which means that in order to have 95 % confidence that a message sent by the source reaches the destination, we need to relay the message through at least five vehicles

of vehicles used to relay any given message. One possible implication of this is privacy; the owner of a vehicle may not want to disclose the vehicle's destination or trajectory information to RSUs. This can be easily resolved as follows: When an RSU offers a message to a vehicle, it also includes the destination information of the message. The vehicle can then reply either "YES" if it can carry the message or "NO" if it cannot carry the message based on its driving trajectory. The vehicle may also choose to reply "Do not Know" if it does not want to disclose any information about its destination; in this case, the original protocol can be used.

In this modified protocol, an RSU will relay a message to the minimum number of required vehicles; sometimes just one vehicle will be enough. However, this makes the protocol more prone to message dropping attacks (a malicious vehicle accepts a message for relaying but does not deliver it to the destination). Possible solutions include the use of end-to-end acknowledgement or an increasing of redundancy by using more than the minimum required vehicles for message relay.

5 Security Support by the Proposed Design

Security is one of the most important requirements of any VANET architecture. Security must be integrated in design from the start. In this section we discuss the security support in our proposed design. We present two different security architectures and other advanced security features such as data verification and secure positioning. The proposed security architectures are specially suited to VANET initial deployment stage.

5.1 Security Architecture

Most currently proposed security architectures assume that all participants of VANET (vehicles as well as roadside units) have certificates [36–42]. The certificates are usually issued by certification authorities that are organized hierarchically on regional basis. Since vehicles are not restricted to any one area and it is not unusual for vehicles to move across regional boundaries, we need elaborated trust relationships among these certification authorities. Further, to ensure privacy, a large number of temporary certificates (or called pseudonyms) are usually used. Protection of these large numbers of valid certificates requires expensive tamper-proof devices (TPDs). Issuance, renewal and revocation of these certificates rely on pervasive roadside infrastructure that will not be available during the initial deployment stages of VANET. In addition, it may be unattractive for an ordinary user to maintain/pay for these certificates (and tamper-proof device) during initial stages when not many services will be available.

Facing these challenges, we propose a security architecture that is suitable for VANET during its initial deployment stages and will be easily transformed to final security architecture when VANET matures.

5.1.1 Internet-Like Public-Key Infrastructure

Current Internet web service relies on a well-established client-server security model and public-key infrastructure, where only servers have certificates and use the Transport Layer Security (TLS) Protocol [53]. The first architecture we propose will directly use the similar infrastructure. In this design, only RSUs have certificates while smart vehicles do not.

Such security design has tremendous advantages. First, we can directly use the mature and secure Internet public-key based protocols in VANET. Second, because RSUs are stationary and used mainly for local areas, certificates for RSUs can be issued very flexibly at local town, or city/county, or state level. Certificates for vehicles, on the other hand, have to be at the national level since vehicles can appear in any place in a country. This feature makes certificate management in our design scalable and economical. For example, certificates can be managed by the department of transportation of a town or city. The renewal of certificates may be accomplished via a security vehicle driving along the road and issuing renewal certificates to each RSU passing by (similar to automated meters used by electricity company such as Progress Energy—Progress, [54]). In later stages of VANET deployment, vehicles may also be issued with certificates, thereby improving security and services.

5.2 Multi-Confidence Level Data Verification

The information given by a vehicle to another vehicle or to an RSU can be false. Lack of V2V and V2I communication will make data verification by vehicles themselves very difficult and time consuming. To solve this security issue, we propose a multi-confidence level data verification scheme by moving the burden of data verification from vehicles to RSUs.

An RSU will collect the information from various sources, analyze the information, assign a certain degree of confidence to the information and then relay the information to vehicles and adjacent RSUs. The degree of confidence can improve if the same information is reported from different initiators (like majority voting). Care is necessary to avoid counting the same message more than once. For example, if a vehicle reports an event to two adjacent RSUs, and later the first RSU also relays the same message indirectly (through another vehicle) to the second RSU, then the second RSU has two messages but it should not increase the level of confidence on the information.

By using the confidence levels, the inherent delay in traditional data correlation approaches can be resolved. An RSU will not wait to receive the same information from a minimum number of vehicles in order to trust it and then broadcast it. Instead, it will relay the urgent information on the first possible opportunity with its current level of confidence. Later, if the confidence level improves, it can always relay the information again with an updated confidence level.

Depending on the source, each piece of information can be assigned with a certain degree of confidence. The sources in decreasing levels of confidence are introduced next.

5.2.1 Level 1

Sensors installed in RSUs will generate the information that has the highest degree of confidence. The collected information can be weather conditions, road conditions, average vehicle speed, traffic congestion, traffic density, etc. There should be some mechanisms to guard against tampering of sensors.

RSUs will report important data to adjacent or affected RSUs. This data can either be sent directly if a direct link exists, or be sent indirectly through vehicles. The sending RSU will add its degree of confidence to the data before signing the message. Since RSUs are tamper proof and a transporting vehicle cannot modify the message signed by RSUs, the message has a high degree of confidence. The only possible attack in this case is message dropping by vehicles, or RSUs being compromised.

5.2.2 Level 2

Information reported by a vehicle equipped with a certificate has the next level of confidence (we do not require each vehicle to have a certificate). The messages will contain original source of information, i.e., whether the information is received from another vehicle or has been observed directly by the vehicle. In case the information is received from another vehicle with a certificate, the original received signed message will be relayed. If the message was received from another vehicle without a signature, the information (including the ID of the initiator vehicle) will be signed by the sending one. In this case, the confidence level of information will be the same as that of the initiator.

5.2.3 Level 3

Information reported by vehicles without certificates will have the lowest degree of confidence. This information can easily be spoofed by an attacker. However, we can use other ways to ensure the trustworthiness of such data, such as correlation of the information with that received from other sources, or correlation of the information with the reported time-location stamp (discussed in next subsection).

5.3 Time-Location Based Secure Positioning

Secure positioning or position verification can be defined as finding/verifying the position of a vehicle with a high degree of confidence. A malicious vehicle may lie about its position for several reasons, e.g., the vehicle might wish to inject false information, avoid detection or avoid liability of its malicious behavior. Secure positioning is also an important aid in data verification, since the relation between information and location of a vehicle can be used to establish the confidence of the information. It also helps in guarding against Sybil and Spoofing attacks [55].

We present a light weight mechanism to tie the position of a vehicle to a particular location at a given time based on RSUs. This theme requires that a vehicle has its certificate. When a vehicle passes by an RSU, it can send a signed time-location stamp request message to the RSU. The RSU, after verifying the signature, issues the time-location stamp. The stamp includes the vehicle's ID, RSU's ID, RSU's location and current time. RSUs can maintain accurate time by GPS or by the NIST time signal broadcasts [56]. The vehicle can carry this signed time-location stamp and later presents it as a proof of its presence at a particular time and location.

If a vehicle does not have its certificate, the above theme can still work by letting an RSU provide the time-location stamp to a passing-by vehicle. In this way, vehicles can still use their obtained stamps as a proof for their past trajectories or locations, but it is not guaranteed to be secure since a stamp is not tied to a particular vehicle anymore without vehicle's digital certificate. Nevertheless, in this case the theme can still provide a certain level of secure positioning service for those non-critical VANET applications.

6 Threats and Their Defenses

A secure design should be capable of thwarting different kinds of attacks launched by malicious nodes. A number of such threats and the safeguards against these threats that are offered by our proposed design are discussed next.

False Information A malicious vehicle may try to pass false information to other vehicles or RSUs. A vehicle with a certificate will sign the message and thus will be liable to the information it generates. If a vehicle does not have a certificate, then it can easily generate false messages without worrying about the liability. However, in this case the damage will be limited since the confidence level attached to such messages will be the lowest.

Further, in our system design, RSUs always have certificates and will sign every message they send out. Thus messages sent from RSUs will not have false information (RSUs are tamper proof). Further, the data verification system defined in the design (discussed in Sect. 5.2) may be used to identify false information initiated from vehicles. The multi-confidence level design enables us to pass time-critical information without compromising on data verification.

Message Dropping A malicious vehicle relaying a message may choose to drop the message. It is very difficult to tell whether message dropping is due to malicious attacks, or due to message delivery failure (e.g., when a message delivery vehicle diverts before passing the destination vehicle or RSU). There are two options to address this problem. One is to enable end-to-end message acknowledgement and the other is to relay the message using more vehicles.

We can extend the analysis in Sect. 4.4.2.3 to determine how many vehicles an RSU should use to deliver a message. Suppose p_{sd} is the probability that a vehicle passing the source will also pass the destination; p_m is the probability that a vehicle passing the destination is malicious and hence drops the message. The effective probability p that a vehicle passing the source will pass the destination and deliver the message will be given by

$$p = p_{sd}\,(1 - p_m) \tag{8}$$

Putting the value of p in Eq. (7), we can compute the minimum number of vehicles to relay a message with a certain confidence. Figure 6 shows the effect on the minimum number of vehicles required to transmit a message when vehicles passing the destination have a probability of $p_m = 0.2$ to be malicious.

Message Modification In such attacks, a malicious vehicle modifies a relaying message. Since all RSUs have certificates and will sign each message, any modification by a relaying vehicle will be detected by the receiving RSU based on the

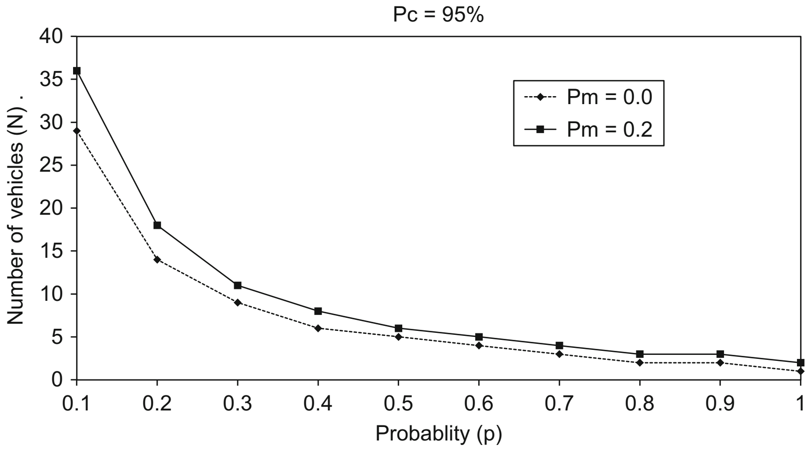


Fig. 6 Number of Relay Vehicles (*N*) required to deliver the message at the destination with a 95 % probability of confidence (*Pc*) for different probabilities (*p*) that a vehicle passing the source will also pass the destination and probability $p_m = 0.2$ that a vehicle passing destination is malicious and drops the message

signed digital signature. Therefore, in our proposed system design, we do not need to worry about message modification attacks.

Spoofing/Impersonation The design does not require each smart vehicle to have certificates, so it is easy for a malicious vehicle to spoof a message. However, as the spoofed message will not have any signature, it will therefore have a very low associated confidence level thus minimizing its potential damage. Spoofing may also be detected if the malicious vehicle has a certificate and the message is not signed.

Privacy Violation/Movement Tracking During normal operation, an information reporting vehicle is not required to reveal its identity, destination or trajectory. Further, our design does not require vehicles to use permanent certificates, thus supporting privacy. A vehicle decides by itself whether to report its time-location stamp to an RSU. In order to track a vehicle, an attacker needs to travel along the vehicle to eavesdrop on the vehicle's time-location request or report messages, which makes such attacks difficult and unattractive to conduct.

Message Replay The major portion of data traffic will be messaging between RSUs and vehicles. Messages sent from RSUs include time-location stamps. Thus, it will not be possible to replay the same message later in time or at a different location. Therefore, our proposed design is secure against message replay attack.

Routing Attacks The protocol mainly relies on the source to decide the routing of a message. Messages transmitted by an RSU include all the routing information and is signed by the RSU to preserve its integrity. Hence, the protocol is secure against different routing attacks. Further, when a new RSU joins the network, it broadcasts its location and identity by signed messages, which are difficult to spoof. The updating of routing tables can even be done manually through special messages from central authority (e.g., sent from passing-by authority vehicles, like—Progress,

[54]), which is manageable since RSUs are not added very frequently. Therefore, our proposed system design is secure against routing attacks.

False Position Reporting A malicious vehicle may falsify its location to avoid liability or to launch an attack. The use of a time-location based secure positioning system introduced earlier can thwart this attack. The position reported by a vehicle without a time-location stamp has a very low degree of confidence. A vehicle not located close to any RSU can report its current position by using its latest time-location stamp, which limits the false position reporting bounds.

Sybil Attacks A vehicle may try to present itself at two or more different locations at any one time, which is called "Sybil attack" [55]. For example, a vehicle can generate a fake congestion message by reporting a large number of vehicles at a location. The time-location based secure positioning guards against this type of attacks. Each time-location stamp has a vehicle's identification; a vehicle, therefore, cannot acquire different time-location stamps from RSUs to launch a successful Sybil attack. Although this type of attack may be possible if vehicles do not have their own certificates, such attack messages will have a low confidence level, and hence, the damage from such attacks is minimal.

Compromised RSUs RSUs are the core and backbone for our design. Thus we specify that RSUs should use temper-proof hardware, which makes them hard to be physically tempered and compromised by attackers. Because RSUs are stationary and always sign with their certificates any messages sent from them, if one RSU is logically compromised by an attacker, it is easily accountable and traceable for all abnormal messages it generates or forwards out. For this reason, a compromised RSU can be quickly detected, located, and removed.

7 Simulations

Simulations were carried out to check the effectiveness of our proposed system. The simulator does not incorporate the details of lower level protocol layers without the implementation of physical and MAC layers. All simulated vehicles and RSUs have the same transmission and reception ranges. A message transfer between a source and a destination is assumed to be successful if both entities are within the communication range of each other.

7.1 Simulation Scenario I

This set of simulations were carried out to find the minimum number of vehicles required to successfully transfer a message from a source RSU to a destination RSU with a given probability of confidence. A region of 25,000 $\times$ 6250 m with road network as shown in Fig. 4 was simulated. When a vehicle traveling towards the destination RSU passes the source RSU, the source RSU transmits the message to the vehicle. The message is then carried by the vehicle for possible delivery to the destination RSU. At each road junction, the vehicle decides to either maintain its direction of travel or divert according to a predefined probability. If the vehicle diverts and hence fails to deliver the message to the destination RSU, the source RSU retransmits the message. This procedure is repeated until the message is successfully received by the destination RSU. In each simulation run, the source RSU sends 1000 messages and the number of retransmissions for each message is recorded. The simulation is repeated 100,000 times and the average number of messages received successfully after a particular number of retransmissions is recorded.

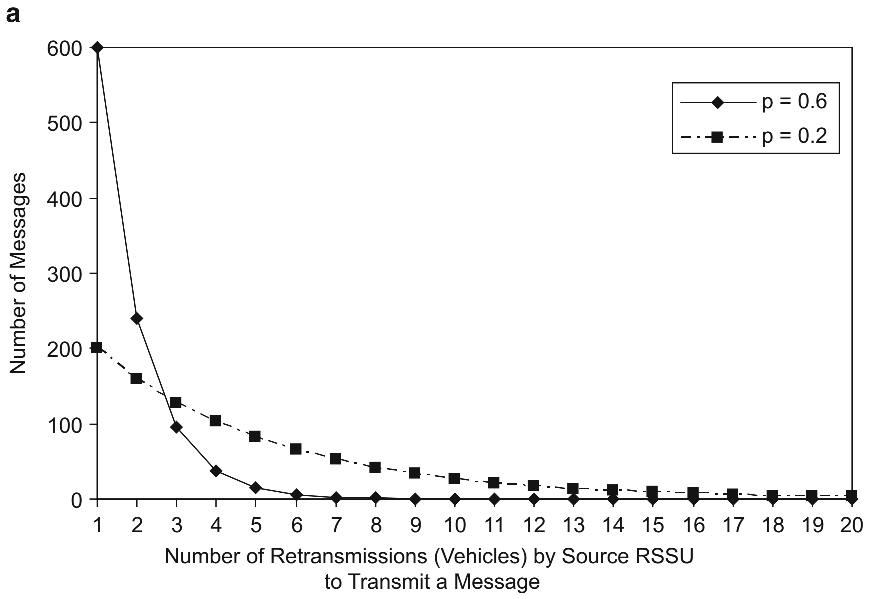
Figure 7a shows the number of messages successfully received (Y-axis) using a particular number of retransmissions (X-axis) for $p = 0.2$ and $p = 0.6$ (p is the probability that a vehicle passing the source will also pass the destination). It shows that, for $p = 0.6$ case, more than 90 % of messages can be successfully received by receiver within four retransmissions. From a different perspective, Fig. 7b shows the number of received messages at the destination after less than or equal to each given number of retransmissions. Figure 7b can be used to find the minimum number of vehicles required to successfully transmit a message with a certain probability of confidence. Figure 7c shows the number of vehicles required for confidence $Pc = 95$ % for each probability p. The simulation results shown in Fig. 7c are identical to the analytical results presented in Fig. 5.

7.2 Simulation Scenario II

During the initial stages of VANET deployment, V2V communication will not be very effective. In addition, due to limited road infrastructure, the V2I communication will also be very limited. This will be a major setback to all VANET applications, such as transfer of a safety message from a point of incident to vehicles entering the area, or information about road blockage for possible diversion.

We have considered two cases and compared the number of vehicles and time required to transfer a message from a source of information (which can be a vehicle passing the scene of incident, or an RSU) to a destination (which can be an emergency response vehicle or an RSU). In the first case, we have a limited roadside infrastructure and messages are transferred between the source and the destination via vehicles only. In the second case, we have intermediate standalone RSUs between the source and the destination, which help in relaying the message. In this case the source is also a standalone RSU. Simulations will help us ascertain the effectiveness of our proposed system in relaying messages using vehicles with or without the intermediate standalone RSUs.

We simulate a region of $25{,}000 \times 6250$ m with a road network as shown in Fig. 3. The number of smart vehicles on the simulation field (a total road length of 35,000 m), at any one time, is kept to five. This small number of vehicles is used to check the effectiveness of our proposed system during the initial deployment stages of VANET. V2V communication is ignored due to this small number of smart



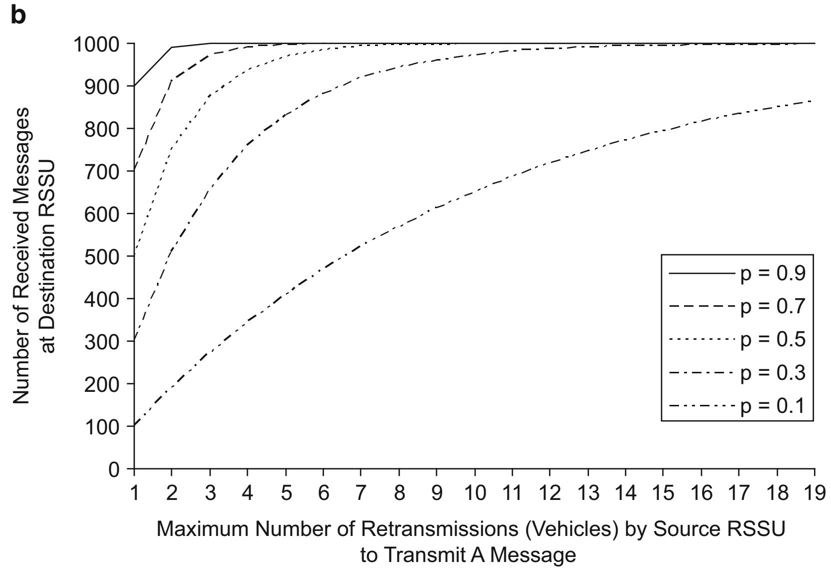


Fig. 7 (**a**) For the probability $p = 0.2$ and $p = 0.6$ (that a vehicle passing the source will also pass the destination), the number of messages successfully received at the destination RSU after a given number of retransmissions by the source RSU. (**b**) For different values of probability p, the number of received messages at the destination after less than or equal to each given number of retransmissions. (**c**) Number of Relay Vehicles (N) required to deliver the message to the destination with a 95 % confidence probability (Pc) and different values of probability (p) [49]

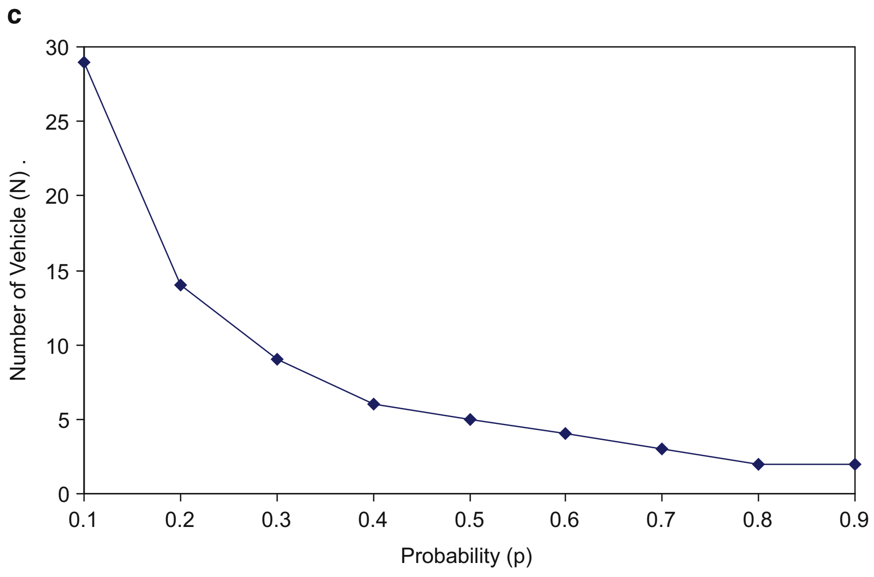
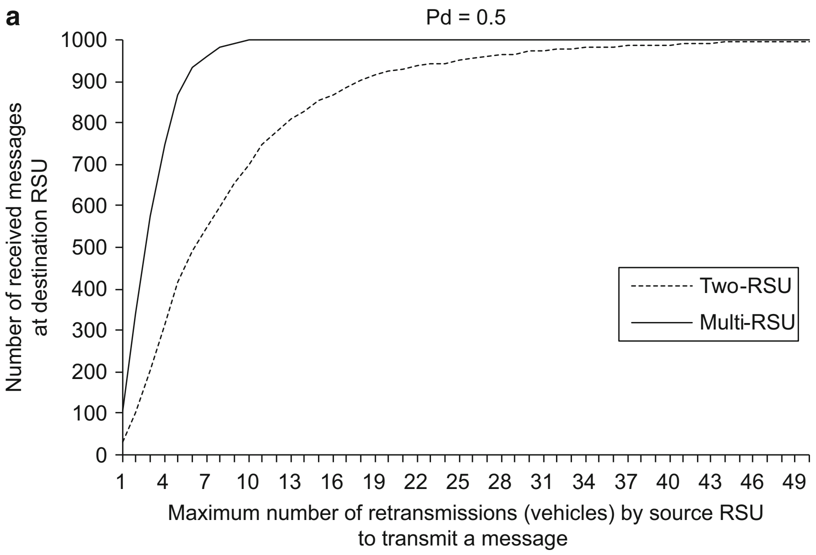


Fig. 7 (continued)

vehicles. At each junction, a vehicle can divert from its current direction of travel with a probability of diversion *Pd*.

In both cases, the source RSU retransmits the message until it is received by the destination. In the second case, a vehicle carrying a message relays it to any intermediate RSU that it encounters. The number of retries (vehicles used to carry the information from the source) and the total time taken for the information to reach the destination is recorded for each message. A total of 1000 messages are transmitted in each simulation. The number of messages received at the destination after less than or equal to each given number of retries for $Pd = 0.5$ are shown in Fig. 8a. The results indicate that the use of multiple (standalone intermediate) RSUs decreases the number of retries considerably.

Figure 8b shows the number of relay vehicles used to transmit the message to the destination with a 95 % of confidence probability. The probability of diversion is varied from 0.1 to 0.9. The results show that for the first case (without intermediate RSUs) the number of vehicles reaches its minimum value when $Pd = 0.5$. This is due to the road layout: at the first road junction a small value of *Pd* is helpful, but at the second road junction a large value of *Pd* is more advantageous. The number of vehicles required for the scenario with multiple RSUs almost remains constant. This happens because the vehicles traveling on other roads also help in the successful delivery of messages. The same pattern of results is obtained in the transmission delay of messages as shown in Fig. 8c. The results indicate a high performance gain when multiple (standalone intermediate) RSUs are used, and the transmission delay will be much more stable than the case when only two RSUs are used. This is true



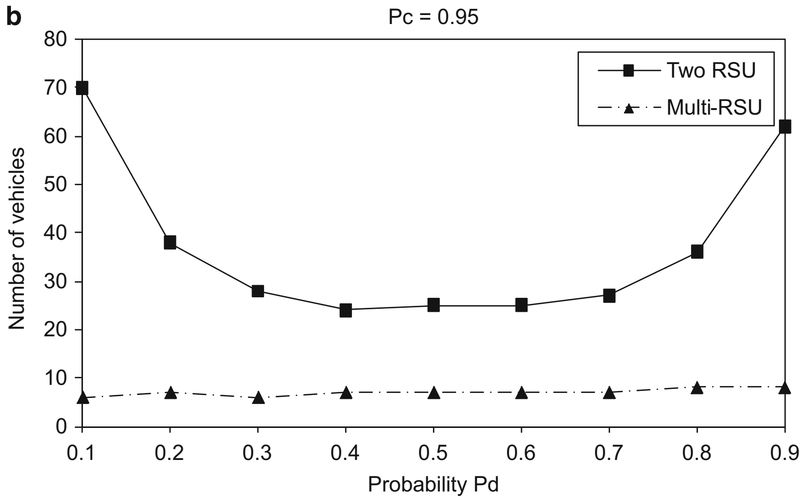


Fig. 8 (**a**) For probability of diversion $Pd = 0.5$ (that a vehicle passing road junction will divert from its direction of travel), the number of received messages at the destination after less than or equal to each given number of retransmissions by the source RSU. (**b**) Number of Relay Vehicles used by the source RSU to deliver the message at the destination with a 95 % probability of confidence (*Pc*) for different probabilities (*Pd*). (**c**) Message transmission delay for different probabilities (*Pd*) [49]

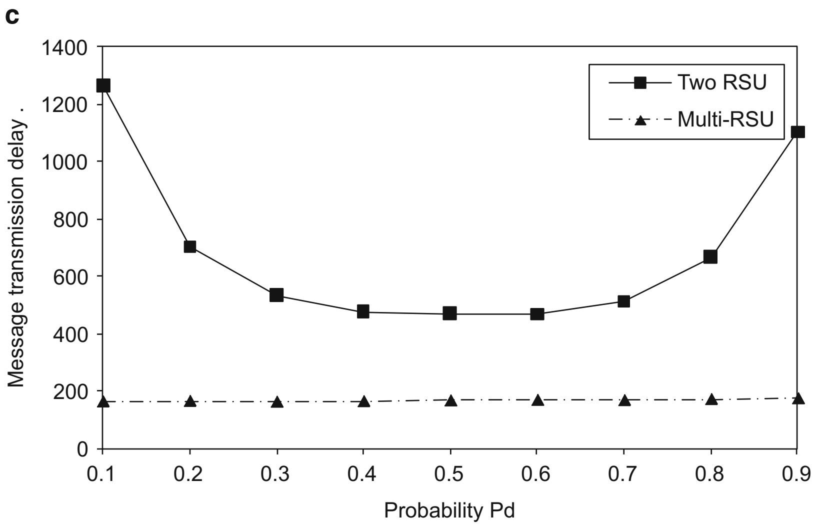


Fig. 8 (continued)

for both the message transmission delays and the number of relay vehicles required for successful message transmission.

8 Discussions

The proposed system effectively meets the challenges highlighted in Sect. 3. Details are given below:

- The proposed system provides an immediate solution to the problem existing during VANET initial deployment stage before a critical mass is achieved.
- The proposed system maintains VANET function in scenarios where V2V communication is not possible due to road layout or traffic conditions.
- The proposed system is progressive. RSUs of varying degrees of functionality can be integrated and later upgraded without the need to an overhaul of existing systems.
- The proposed system is an economical solution.
- The proposed system exhibits good scalability. More areas can easily be included in an existing VANET network by simply adding more RSUs. In addition, initially isolated regions can be later interconnected by RSU to RSU links.
- The minimum number of RSUs required for the proposed system to work is very small as compared to conventional solutions.

- The proposed system does not require an elaborate certification system to ensure security. The presented security architecture provide the necessary security with minimal requirements of infrastructure.
- The proposed system provides data verification and secure positioning services with the help of RSUs.

There are some limitations in the proposed design. First, because communication relies on RSUs to relay, it may be slow for vehicles to receive time-critical messages compared with V2V (or V2I with I2I) communication. However, in the VANET initial transition period, V2V and also I2I communication might not be possible due to the low density of smart vehicles on the roads and a lack of fully networked roadside units. Second, the VANET communication relies on the RSU infrastructure. It is possible that in some rural areas there are no RSU devices installed. Third, RSU to RSU indirect communication relies on passing-by vehicles. Thus the communication may be slow and can be interrupted frequently when there are few smart vehicles around.

9 Conclusions

There are numerous proposed applications of VANET but most of them are not practical until a critical mass of fully networked roadside units and smart vehicles is achieved. It will be very difficult to achieve this critical mass in the initial years of VANET deployment. This difficulty will further slow down the market penetration. In this chapter, we have presented an economical and practicable solution to address this issue, which incorporates and relies on a very limited numbers of roadside units with very basic functionalities. Our solution is economical, scalable and upgradeable. We show that the solution is practical with the help from a small number of smart vehicles. The future work includes use of real traffic data for simulations and experiments. We plan to carry out real traffic monitoring to obtain traffic statistics and data on high-end vehicles (these vehicles possibly will be the first in installing wireless devices) to study possible market penetration. We will obtain such data either from the department of transportation, or through our manual observation of real road traffic.

References

1. Lee, C.-H., Huang, C.-M., Yang, C.-C., Wang, T.-H.: Co-SVC-MDC-based cooperative video streaming over vehicular networks. Comput. J. **55**(6), 756–768 (2012)
2. Agarwal, A., Starobinski, D., Little, T.D.C.: Exploiting downstream mobility to achieve fast upstream message propagation in vehicular ad hoc networks. INFOCOM/MOVE (2007)
3. Jain, S., Fall, K., Patra, R.: Routing in a delay tolerant network. SIGCOMM'04 (2004)
4. Fall, K.: A delay-tolerant network architecture for challenged internets. SIGCOMM'03 (2003)

5. UMass DiselNet. [online] Available at http://prisms.cs.umass.edu/dome/umassdieselnet (2011). Accessed 25 Oct 2011
6. Bechler, M., Franz, W.J., Wolf, L.: Mobile internet access in FleetNet. 13th Fachtagung Kommunikation in verteilten Systemen (2003)
7. Santa, J., Moreo, R.T., Skarmeta, A.F.G.: A novel vehicle communication paradigm based on cellular networks for improving the safety in roads. Int. J. Intell. Inf. Database Syst. **2**(2), 240–257 (2008)
8. Ko, Y.F., Sim, M.L., Nekovee, M.: Wi-Fi based broadband wireless access for users on the road. BT Technol. J. **24**, 122–129 (2006)
9. Wu, H., Fujimoto, R.M., Guensler, R., Hunter, M.: MDDV: mobility centric data dissemination algorithm for vehicular networks. VANET'04 (2004)
10. Xu, Q., Mak, T., Sengupta, R.: Vehicle-to-vehicle safety messaging in DSRC. VANET'04 (2004)
11. Korkmaz, G., Ekici, E., Ozguner, F., Ozguner, U.: Urban multi-hop broadcast protocol for inter-vehicle communication systems. VANET'04 (2004)
12. Vahdat, A., Becker, D.: Epidemic routing for partially-connected ad-hoc networks. Technical Report CS-2000-06. Duke University (2000)
13. Wisitpongphan, N., Bai, F., Mudalige, P., Sadekar, V., Tonguz, O.: Routing in sparse vehicular ad hoc wireless networks. IEEE J. Sel. Areas Commun. **25**(8), 1538–1556 (2007)
14. Wisitpongphan, N., Tonguz, O., Bai, F., Mudalige, P., Sadekar, V.: On the routing problem in disconnected vehicular networks. INFOCOM'07 (2007b)
15. Spyropoulos, T., Psounis, K., Raghavendra, C.S.: Efficient routing in intermittently connected mobile networks: the multi-copy case. IEEE Trans. Netw. **16**(1), 77–90 (2007)
16. Mabiala, M., Busson, A., Vèque, V.: Inside VANET: hybrid network dimensioning and routing protocol comparison. VTC'07-Spring (2007)
17. Little, T., Agarwal, A.: An information propagation scheme for VANETs. ITSS'05 (2005)
18. Hull, B., Bychkovsky, V., Zhang, Y., Chen, K., Goraczko, M., Miu, A., Shih, E., Balakrishnan, H., Madden, S.: CarTel: a distributed mobile sensor computing system. SenSys'06 (2006)
19. Goodman, D., Borras, J., Mandayam, N., Yates, R.: INFOSTATIONS: a new system model for data and messaging services. VTC'97 (1997)
20. Small, T., Hass, Z.J.: The shared wireless infostation model – a new ad hoc networking paradigm. MobiHoc'03 (2003)
21. Wyatt-Millington, R.A., Sheriff, R., Hu, Y.F., Conforto, P., Losquadro, G.: The SUITED project: a multi-segment system for broadband access to Internet services. IEEE Broadband Satellite Conference (2000)
22. iCartel: MIT CarTel. [online] Available at http://icartel.net/icartel-docs/. Accessed 24 Oct 2011
23. Gutiérrez, J.: Selected readings on telecommunications and networking. Idea Group Inc (IGI) (2008)
24. Data Calculator. [online] Available at http://www.att.com/standalone/data-calculator. Accessed 24 Oct 2011
25. Luo, J., Hubaux, J.-P.: A survey of research in inter-vehicle communications. In: Securing Current and Future Automotive IT Applications, pp. 111–122, Springer (2005)
26. Levacher, K., McGee, F., Murphy, F.: A comparison between 3G and 802.11 wireless technologies for Inter-Vehicular Communications purposes, [online] Available at http://killian.levacher.googlepages.com/Acomparisonbetween3Gand802.11wireles.pdf (2007). Accessed 24 Oct 2011
27. Qureshi, A., Guttag, J.: Horde: separating network striping policy from mechanism. In: 3rd International Conference on Mobile Systems, Applications, and Services (2005)
28. Chakravorty, R., Clark, A., Pratt, I.: GPRSWeb: optimizing the web for GPRS Links. In: ACM/USENIX MobiSys, San Francisco (2003)
29. Chan, M.C., Ramjee, R.: TCP/IP performance over 3G wireless links with rate and delay variation. In: ACM Mobicom (2002)

30. Zhao, J., Cao, G.: VADD: vehicle-assisted data delivery in vehicular ad hoc networks. INFOCOM'06 (2006)
31. Lochert, C.S, Caliskan, B.M., Mauve, M.: The feasibility of information dissemination in vehicular ad-hoc networks. WONS'07 (2007)
32. Petit, B., Ammar, M., Fujimoto, R.: Protocols for roadside-to-roadside data relaying over vehicular networks. WCNC'06 (2006)
33. Ding, Y., Wang, C., Xiao, L.: A static-node assisted adaptive routing protocol in vehicular networks. VANET'07 (2007)
34. Mansy, A., Ammar, M., Zengura, E.: Reliable roadside-to-roadside data transfer using vehicular traffic. MASS'07 (2007)
35. Chuah, M.C., Fu, F.: Performance study of robust data transfer protocol for VANETs'. LNCS, vol. 4325, pp. 377–339. Springer, Berlin (2006)
36. Raya, M., Papadimitratos, P., Hubaux, J.-P.: Securing vehicular communications. IEEE Wirel. Commun. Mag. **13**, 8–15 (2006)
37. Papadimitratos, P., Buttyan, L., Hubaux, J-P., Kargl, F., Kung, A., Raya, M.: Architecture for secure and private vehicular communications. International Conference on ITS Telecommunications (2007)
38. IEEE Std 1609.2: IEEE trial-use standard for wireless access in vehicular environments – security services for applications and management messages, pp. 0_1–105 (2006)
39. Armknecht, F., Festag, A., Westhoff, D., Zang, K.: Cross-layer privacy enhancement and non-repudiation in vehicular communication, WMAN'07 (2007)
40. Fan, C.I., Hsu, R.H., Tseng, C.H.: Pairing-based message authentication scheme with privacy protection in vehicular ad hoc network. International Conference on Mobile Technology, Applications and Systems (2008)
41. Crescenzo, G.D., Zhang, T., Pietrowicz, S.: Anonymity notions for public-key infrastructures in mobile vehicular networks. MASS'07 (2007)
42. Choi, J., Jung, S.: A security framework with strong non-repudiation and privacy in VANETs'. CCNC'09 (2009)
43. Papadimitratos, P., Mezzour, G., Hubaux, J.P.: Certificate revocation list distribution in vehicular communication systems. VANET'08 (2008)
44. Stampoulis, A., Chai, Z.: Survey of security in vehicular networks, [online] project CPSC 534, Available from http://zoo.cs.yale.edu/~ams257/projects/wireless-survey.pdf (2007). Accessed 25 Oct 2011
45. IEEE Std 802.16: IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems. IEEE Std 802.16-2009 (Revision of IEEE Std 802.16-2004), pp. C1–2004 (2009a), 29 May 2009
46. IEEE Std 802.16j: IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems Amendment 1: Multiple Relay Specification. IEEE Std 802.16j-2009 (Amendment to IEEE Std 802.16-2009), pp.c1–290 (2009b), 12 June 2009
47. IEEE Std 802.16h: IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems Amendment 2: Improved Coexistence Mechanisms for License-Exempt Operation. IEEE Std 802.16h-2010 (Amendment to IEEE Std 802.16-2009), pp. 1–223 (2010), 30 July 2010
48. IEEE Std 802.16m: IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems Amendment 3: Advanced Air Interface. IEEE Std 802.16m-2011(Amendment to IEEE Std 802.16-2009), pp. 1–1112 (2011), 5 May 2011
49. Aslam, B., Wang, P., Zou, C.C.: An economical, deployable and secure vehicular ad hoc network. In: Proceedings of IEEE Military Communications Conference (MILCOM'08), San Diego, 17–19 November 2008
50. Biswas S., Morris, R.: ExOR: opportunistic multi-hop routing for wireless networks. SIGCOMM'05 (2009), August 2005

51. Kim, J., Bohacek, S.: A comparison of opportunistic and deterministic forwarding in mobile multihop wireless networks. MobiOpp'07 (2007)
52. Jacobson, V.: Congestion avoidance and control. In: Proceedings of the ACM SIGCOM'88 Conference, pp. 314–329, August 1988
53. Dierks, T., Rescorla, E.: RFC 4346: The transport layer security (TLS) protocol version 1.1 (2006)
54. Progress Energy to install new automated meters for residential customers, [Online] Press Release, Available at https://www.progress-energy.com/company/media-room/news-archive/press-release.page?title=Progress+Energy+to+install+new+automated+meters+for+residential+customers+&pubdate=03-30-2005 (2005). Accessed 25 Oct 2011
55. Douceur, J.: The Sybil attack. In: First International Workshop on PeertoPeer Systems (2002)
56. Lombardi, M.A.: National institute of standards and technology (NIST) Special Publication 432'. NIST Time and Frequency Services, Edition 2002 (revised April 2003)

Deception-Based Survivability

Ruchika Mehresh and Shambhu J. Upadhyaya

Abstract Critical systems in current threat landscape demand more than just fault-tolerance and security; they demand survivability. Survivability is the ability of a system to continue delivering essential services during attacks, faults or accidents. This is usually accomplished via a four-layered defense setting that consists of prevention, detection, recovery and adaption. As evident by the recent incline in advanced persistent threats, the existing defense systems are in a dire need of evolution. This new generation of defense systems should be smart, adaptive and unlike traditional systems, stay ahead of the malicious actors. Many researchers have started to consider deception as a means to this end. Deception involves misrepresenting or hiding information in order to manipulate a user's actions. When engrafted in the prevention and detection layers, deception can help trace *attacker intent, objective and strategies (AIOS)* which aids in the development of targeted recovery and adaptation procedures. Such procedures, in turn, help a system survive in hostile environments. Though the adoption of deception-based defense has been hindered by legal and moral issues in the past but the recent increase in interest in this field holds great promise. This chapter discusses deception-based survivability, its benefits and shortcomings. It presents a high-level deployment architecture that uses deception to ensure system survivability. Other aspects of deception-based survivability such as performance overhead, continued effectiveness and precision have also been discussed.

1 Introduction

Cyber-warfare is no longer limited to military domain. Knapp and Boulton reviewed information warfare literature from 1990 to mid-2005 and described how cyber warfare has extended to other domains outside military [16]. Mission-critical systems run crucial applications and hence, become the most likely targets of security attacks. Therefore, critical systems in both military and industrial domains

R. Mehresh • S.J. Upadhyaya (✉)
University at Buffalo, State University of New York, Buffalo, NY 14260, USA
e-mail: ruchika.mehresh@gmail.com; shambhu@buffalo.edu

need mission assurance. Mission Assurance is an umbrella term that includes principles to be followed through a mission's life-cycle to ensure its success. It is all about proactively identifying and mitigating deficiencies in mission design, its deployment and its operation to prevent failures.

Baskerville [3] discussed *asymmetric warfare* and how it applies to information warfare. Asymmetric warfare theory states that attackers have the advantage of time and stealth over defenders. In order to counter this imbalance, defense needs to be "agile and adaptive." The emergence of *advanced persistent threat (APT)* has further added to the challenge of satisfying today's mission assurance needs. APTs are usually characterized by extreme stealth, advanced skill-set, vast resources and hence, a markedly high success rate.

Mission assurance usually stresses on providing fault-tolerance and security—two contradictory notions. While security requires limited trusted computing base, fault-tolerance demands replication and redundancy. A three phased-approach, consisting of prevention, detection and recovery, is usually employed to satisfy these conflicting requirements. The idea of survivability goes beyond mission assurance. Ellison et al. [10] describe survivability as "the capability of a system to fulfill its mission in a timely manner in the presence of attacks, failures and accidents." Survivability focuses on the continuity of a mission without relying on the guarantee that precautionary measures will always succeed. It concentrates on the impact of an event rather than its cause. Survivability requires four basic layers of protection:

- Prevention or resistance against faults/attacks.
- Detection of faults/attacks.
- Full recovery of the essential services (mission) after the fault/attack.
- Adaptation or evolution to reduce the possibility or effectiveness of future faults/attacks.

While the first two layers, prevention and detection, already provide strong defense, recovery is the fallback plan should these layers fail to protect the system. However, Mehresh et al. discuss how the recovery layer can be attacked and present a brief survey of many possible attacks on the various recovery schemes [26]. Therefore, recovery being the last phase, needs protection (or a fallback) as well. Because adaptation/evolution mechanisms are generally activated during or after the recovery, they are rarely effective if recovery fails. Despite the advances in mission survivability, the existing solutions by far remain ineffective against APTs.

One of the major challenges of designing a mission survivable system is to ensure that all the precedented or unprecedented threats can be detected, while conserving the timeliness property of the mission. Because dealing with unprecedented attacks (zero-day attacks) requires monitoring the entire traffic, it becomes difficult to ensure the timeliness property. Hence, surviving unprecedented threats and conserving the timeliness property are two conflicting requirements. We believe that deception is an effective tool in handling this conflict and evening out the asymmetry in cyber warfare.

Defensive deception is an act of intentional misrepresentation of facts to make an attacker take actions in defender's favor [9]. The framework presented in this chapter leverages concepts of deception in a hardware-based security setup [6, 25, 44]. It adapts its behavior based on the observations on the system. If suspicious activity is detected but imminent danger is not perceived, it does not raise alerts or activate recovery procedures. Instead, it continues to behave normally and observe to better understand *attacker intent, objective and strategies (AIOS)*. This approach has two advantages. First, it assists in the designing of targeted recovery procedures which are lightweight and highly effective. Second, providing incomplete information to an attacker (by not raising alerts or recovering immediately) prevents any harmful repercussions. Instead of recovering right away, the system relies on replication to account for the resulting component unreliability. Replication provides reliable alternatives to suspected (but unrecovered) system components. Although costly, redundancy and replication are requirements for the fault-tolerance aspect of mission survivability. We are merely extending its use to address security.

Deception-based survivability solution is in a way, a game changer. Instead of relying on a generic mix of traditional solutions, it assists in the design of a more effective (targeted) recovery in response to an advanced attack. In case of stealth attacks, it buys the system time to figure out AIOS while the attacker remains oblivious of any detections. Note that no current or prior survivability solutions work in this manner. The general rule is to detect anomalies/attacks and raise alerts that initiate recovery. However, if imminent danger is not perceived, the recovery can be delayed and unreliability can be covered using replication to buy time to completely profile an attack and earn all the aforementioned advantages.

The main contribution of this chapter is the design and development of a deception framework to address the scourge of advanced persistent threats that are prevalent in today's high value systems and mission-critical applications. This framework is developed by studying the various deception schemes in the literature and then deriving a set of axioms to help with the formal development of this framework. We present a detailed background on deception and APT in Sect. 2. Section 3 models an attack flow based on APT. From this model, formal requirements for next-generation survivability solutions are derived in Sect. 4. These requirements serve as a basis for the deception-based survivability solution presented in Sect. 5. Finally, the chapter ends with a detailed discussion and conclusion in Sect. 6.

2 Background

2.1 Deception

Deception itself in warfare is not new [6, 42]. However, deception has associated legal and moral issues with its usage in today's society. Cohen, the author of deception toolkit [5] discusses moral issues associated with the use of deception throughout his work [6]. Lakhani discusses the possible legal issues involved in the use of deception-based honeypots [17].

Deception aims to influence an adversary's observables by concealing or tampering information. Murphy discusses the techniques of deception such as fingerprint scrubbing, obfuscation, etc. [28]. Her work is based on the principle of holding back important information from the attacker to render the attack weak. There is vast literature and taxonomies on the use of deception to secure computer systems and information in general [6, 36, 37].

The static nature of today's networks presents a sitting and vulnerable target. Moreover, patch development time for most exploits is much higher than the exploit development time, further putting critical systems at a disadvantage. Repik documents a summary of internal discussions held by Air Force Cyber Command staff in 2008 [36]. His work makes a strong argument in favor of using deception as a tool of defense. He discusses why planned actions taken to mislead hackers have merit as a strategy and should be pursued further.

2.2 Advanced Persistent Threats

Daly describes APTs as sophisticated cyber-attacks by hostile organizations that aim to gain access, maintain a foothold and modify data at their target systems [8]. The term *'advanced'* refers to the high-quality skill set involved in designing these attacks. The term *'persistent'* is used to indicate the long presence of an attacker inside the system (for purposes such as spying or stealthy privilege escalations). APTs are "a new breed of insidious threats that use multiple attack techniques and vectors and that are conducted by stealth to avoid detection so that hackers can retain control over target systems unnoticed for long periods of time" [41].

Existing market forces and easy access to high-end technology have considerably altered the cyber attack landscape. As reported by Washington Post, malicious sleeper code is known to be left behind in the U.S. critical infrastructure by state-sponsored attackers [29]. This sleeper code can be activated anytime to alter or destroy information. Similar stealth methodologies are also employed during multi-stage delivery of malware discussed in [35] and the botnet's stealthy command and control execution model in [15].

We already see a rising trend of stealthy and advanced malware all around [14]. Let us review some major recent incidents of APTs that had a widespread impact:

- **Stuxnet:** Stuxnet sniffs for a specific configuration and remains inactive if it is not found [11]. "Stuxnet is the new face of twenty-first century warfare: invisible, anonymous, and devastating" [13]. Since it came to light in 2010 when it targeted Iran's nuclear facilities, many variants of this malware have been discovered. They are the most serious threats to SCADA systems such as power plants and gas pipelines [20]. They have the capability to reprogram programmable logic controllers (PLCs) to work to an attacker's advantage and maintain stealth.

- **Flame:** Flame outperformed Stuxnet in sophistication and capabilities. It installs itself at endpoints and sniffs information. This information is then transmitted back to the *command and control (C&C)* center. Based on the information received, C&C decides its attack strategies. Flame spreads slowly and stealthily and hides its traces to ensure that it can get valuable information without raising red flags. It has the capability to monitor nearly every activity at an endpoint including conversations utilizing Bluetooth technology. Flame's operators are highly skilled, stealthy, focused and adapt their attacks to each target [27].
- **Operation Aurora:** Aurora targeted more than 35 large-scale organizations including Yahoo, Symantec, Morgan Stanley, etc. and stole intellectual property. Aurora's multi-phased attack started with advanced social engineering and targeted emails that hosted malicious Javascript code. It mostly exploited zero-day vulnerabilities in Internet Explorer and established backdoor communication with its C&C centers via TCP port 443. It used encryption for stealth in order to avoid traditional detection measures. Since then, many similar advanced attacks have been reported by organizations such as Sony, Barracuda Networks, RSA security, etc. [41]. The most important features of Aurora is its complexity, sophistication and stealth [21]. The installation and working of this malware is completely hidden from the system user.
- **Gh0stNet:** Gh0stNet is an intelligence gathering operation that uses a Trojan horse named Gh0st Rat. Some of its features are key logging, remote terminal shell, tracking videos remotely, voice monitoring, session management, hiding traces by clearing up the systems logs, etc. It has been known to compromise at least 1,295 computers [8].

A report published in 2011 by McAfee surveyed 200 IT executives from critical infrastructure enterprises in 14 countries [2]. The report documents cyber-security experts expressing concerns about the surveillance of U.S. critical infrastructure by other nation-states. The increasing penetration of critical infrastructure by APTs has been a reason to worry for many nation-states [2, 29].

Stealth is usually the underlying feature in APT. It is essential to the lengthy process involved in compromising a highly secured system. In essence, stealth buys attackers time to slowly progress towards their goals by reducing the risk of detection. For instance, some malware delay their activation so a pattern recognition-based anti-virus cannot associate them with their source. Another example is a compromised system contacting its C&C by embedding information (e.g., keystrokes) in the DNS (domain name system) packets [8]. Stealth in malware distribution is generally achieved via junk insertion, code recording and packing [12].

More often than not, APTs are characterized by resourceful, adaptive and stealthy initiators. Note that aggressive attackers are easier to spot and hence, routine security measures are able to take care of them. Their aggressive nature renders their attacks one-shot which means that they usually get detected and blocked during the initial attempts. However, stealth attacks are multi-phased. They use deception to hide from detection while gradually gaining more privileges and information about the system. Such attacks are extremely dangerous and need innovative defense [35].

Advanced attackers usually adapt and alter their strategies based on observations. These changing strategies are executed via C&C architectures. The attacks are targeted and multi-phased. Generally, the first step is to gather intelligence such as the anti-viruses running on the system. This information helps exploit weaknesses in the defense while avoiding detection. Attackers usually establish multiple malware installations and footholds inside the system. Even if a system recovers from some malware installations, there is no guarantee that the system is fully-recovered or attacker-free. Therefore, system administrators need a complete picture of the situation if they intend to recover a system and avoid a generic solution like secure reboot. Rebooting can easily disrupt a mission's continuity and timeline. Frequent rebooting can even lead to a denial of service (DoS) attack. Additionally, if system is restored to the same state that was earlier compromised, there are no guarantees that it will not be compromised again.

An advanced attacker can compromise the deployed security solutions on a host if he acquires the necessary privileges [24]. Thus, strong security is not achievable without strong tamper-resistance. As discussed above, detecting and recovering from a subset of attacker's footholds in the system is not effective. It can even lead the adversaries to become aggressive and advance their attack timeline (based on the game-theoretic principles of risk and reward). They may also try to erase the attack traces by either deleting important files or corrupting the system state, thereby foiling the mission. Such behavior works against the mission survivability requirements. Therefore, any solution designed for mission survivability must be tamper-resistant and must not divulge any information to an attacker prior to considering and analyzing all the repercussions.

Stealthy attackers that hide in the system for long usually possess high privileges and sensitive system information. Therefore, it is best to identify all infiltration efforts during their early stages. The challenge is to correlate isolated events of low-severity suspicious activity to understand the AIOS. This is especially true if these events occur on different systems in a distributed environment. Moreover, once an infiltration attempt is identified, it is hard to determine the extent of system compromise. Thus, as inferred above, any action must be taken after considering its repercussions.

Traditional detection and recovery measures have proven ineffective against APTs. Thus, researchers recommend a combination of diverse security measures to counter it [7]. For instance, Tankard recommends monitoring the outgoing traffic in combination with other traditional security measures (e.g., keeping the system up-to-date, using firewall, data encryption, security audits, etc.) [41]. Good security practices and awareness are always recommended [39]. However, multiple security measures when combined can become heavyweight and harm system's performance, delay a mission or simply fail to scale. They are not even guaranteed to stay effective.

3 Modeling Advanced Persistent Threats

Based on the discussion in Sect. 2, we present a generic APT-based attack flow that can be used in the designing of next-generation survivability solutions. It is an extension of the basic attack flow presented by Repik [36]. The attack flow is described in Algo 1. Let ϕ be the set of exploitable vulnerabilities for a system with state s(t), where t is time. For each vulnerability ν in ϕ, the amount of resources required to exploit it is represented by r[ν]. Total resources available to an attacker is $\hat{r}$. Risk associated with exploiting each vulnerability ν is $\rho[\nu]$. Maximum risk that the attacker can afford is $\hat{\rho}$.

Algorithm 1 Attack pattern for sophisticated attacks

```
1:  while TRUE do
2:      while $\phi$ = NULL AND $\forall\nu,\rho[\nu]\geq\hat{\rho}$ do
3:          Gather intelligence
4:          Develop exploits
5:          Perform network reconnaissance
6:          Update vulnerability set $\phi$
7:      end while
8:      if $\exists\nu$, (r[$\nu$]$\leq\hat{r}$ AND $\rho[\nu]\leq\hat{\rho}$) then
9:          Install backdoors; Update $\hat{r}$
10:         while s(t) $\neq$ ATTACKDISCOVERED do
11:             if s(t) $\neq$ CRUCIALSTAGE then
12:                 WAIT
13:             else if $\exists\nu$, (r[$\nu$]$\leq\hat{r}$ AND $\rho[\nu]\leq\hat{\rho}$) then
14:                 Attack and exploit $\nu$; Update $\hat{r}$; Assess damage
15:                 if s(t)=COMPROMISED then
16:                     Operation successful and Exit
17:                 end if
18:             else
19:                 Terminate operation
20:             end if
21:         end while
22:         if Contingency plan exists then
23:             Execute contingency plan
24:         else
25:             Terminate operation
26:         end if
27:     else
28:         Terminate operation
29:     end if
30: end while
```

A sophisticated attack usually starts with intelligence gathering and initial planning. Based on the available resources, an attacker decides whether to exploit a currently known vulnerability or search for more. Attack occurs in multiple stages such as reconnaissance, installing backdoors or rootkits, developing exploits, etc., until a crucial stage is reached. An attack during crucial stage of the mission has the maximum pay-off for the attacker. If discovered, most advanced attacks have a contingency plan that may involve deleting or destroying system information to hide traces. Note that most APTs are considered to be state-sponsored attacks due to the huge resource and skill investment that they require. Therefore, maintaining stealth becomes an even higher priority for them.

4 Formal Requirements

In light of the threat assessment presented in the previous section, we now list down requirements for a state-of-the-art deception-based survivability framework for mission-critical systems.

- **Prevention**: Prevention is generally the first step towards developing an effective survivability solution. It not only attempts to prevent the attacks but also dissuades attackers with limited resources.
- **Detection**: We identify two main challenges in the detection of APTs. First, the survivability solution should force or manipulate a stealthy attacker into leaving a discernible and traceable pattern. Second, detection of such a pattern should be hidden lest the attacker could get spooked and execute a contingency plan for which the defender may not be prepared.

  For a given system with state $s_1(t)$, there is a set ϕ_1 of suspicious actions (for instance, a possible exploit attempt). A user that chooses an action from this set is malicious with a probability p. This means that he could be benign with a probability 1-p. Let system states $s_1(t)$, $s_2(t)$,....,$s_n(t)$ (where, n is the total number of system configurations) have ϕ_1, ϕ_2,....,ϕ_n as their respective sets of suspicious actions. Some actions are malicious with a higher probability p_i where, $1 \leq i \leq n$. Frequently choosing actions with higher probability p_i indicates the malicious nature of the actor. In a deception-based survivability solution, the defender can choose states with actions that have higher p_i's, which means that if a user keeps choosing the actions from the set ϕ, his probability of being malicious ($p_1.p_2.p_3....p_n$) will cross the threshold quickly. It is like offering vulnerabilities to unsuspecting users and observing if they exploit them. Thus, choosing and controlling these states is crucial in determining if a user is malicious with a higher probability in a shorter time. Note that this sort of detection is not applied to all users because it is resource-intensive and time-consuming. Instead, it is used to quickly refine the AIOS of users that are already the suspects.
- **Effective recovery with adaptation**: If the attacker has penetrated a system by exploiting vulnerabilities, recovering the system to the same old state does not get rid of the vulnerabilities or make the system any more secure. Therefore, the need is to ensure that during each recovery, vulnerabilities that are being exploited are patched. It is usually considered a good practice to employ proactive recoveries

(periodically scheduled) for critical systems. It is much easier to predict the timing impact of proactive recoveries and hence conserve the timeliness property of a survivable system. Reactive recoveries, if evoked excessively, can harm system's performance and mission's survivability. If reactive recoveries are employed, they should have a ceiling for how frequently they can be invoked and for how long each recovery can run.

Note that no prevention, detection or recovery measures should violate the timeliness property of a survivable mission.

- **Zero-day attacks**: Considering that most APTs exploit zero-day attacks, a good survivability solution must be effective in dealing with them. Several anomaly-based detection systems have been proposed in order to detect such attacks [4]. However, Liu et al. describe the big challenge "how to make correct proactive (especially predictive) real-time defense decisions during an earlier stage of the attack in such a way that much less harm will be caused without consuming a lot of resources?" [19]. Schemes that attempt to detect zero-day attacks usually take one of the two approaches: predictive or reactive. Under the predictive approach, all the suspected policy violations are taken as a sign of intrusion. This results in a higher rate of false alarms and hence service degradation. Under the reactive approach, defender takes an action only when he is somewhat sure of the foul play. Generally, it is difficult to know when to react. If the system waits until a complete attack profile emerges, it may be too late to react. A good trade-off is offered using honeypots (a form of deception). The defender redirects all the suspicious traffic through honeypots which are responsible for blacklisting/whitelisting the traffic flows [31, 32]. Researchers have already proposed ways to employ honeynet in a production-based environment [17, 18].
- **Conserving timeliness property**: Timeliness property describes the capability of a mission to stick to its originally planned schedule. This being said, a schedule can account for periodic recoveries and some unexpected delays due to miscellaneous factors. In order to conserve this property, it is essential that all indeterministic time-consuming operations be moved out of the mission's critical path.
- **Non-verifiable deception**: A good deception should be non-verifiable [30]. Deception is difficult to create but easier to verify. For instance, when an attacker attempts to delete a file, even though a deceptive interface can give a positive confirmation of the deletion, the attacker can always verify if the file still exits.

  For a state s(t), an action χ is expected to have an effect ω. Generally, deception (like in honeypots) involves confirming that χ has been performed but the effect ω is never reflected in the system. If the attacker has a feedback loop to verify ω, a deception can be easily identified. Therefore, either the feedback loop needs to be controlled so as to give the impression that ω exists, or the feedback loop should be blocked for all regular users. An open and honest feedback loop can help attacker to figure out ways around deception by trial-and-error.

5 Deception-Based Survivability Framework

5.1 Basics

Preventive deception is the first step in mission survivability. Some traditional preventive measures are firewalls, encryption techniques, access controls, etc. These measures have proved to be very successful in deterring weak adversaries. However, strong and determined adversaries are always known to find their way around these. McGill suggests that the appearance of a system being an easy or a hard target determines the probability of attacks on it [22]. Based on similar literature, we categorize deception-based prevention methodologies under following four headings:

- *Hiding*: Hiding is the most basic form of deception. One could use schemes such as fingerprint scrubbing, protocol scrubbing, etc. to hide information from an attacker [38, 43]. Such schemes can also be used to feed false information to the adversaries. Yuill et al. have developed a model for understanding, comparing, and developing methods of deceptive hiding [44].
- *Distraction*: McGill demonstrates that given two targets of equal value, an attacker is more likely to attack the target with lesser protection [22]. However, Sandler and Harvey analytically prove that this tendency continues only until a threshold. If more vulnerabilities are introduced to a system, an attacker's preference for attacking that system does not increase beyond a certain threshold. System observables that attackers rely on can be manipulated to feed misinformation or hide information from attackers. Thus, strategies can be devised to affect an attacker's perception of the system. There are studies that model threat scenarios based on target's susceptibility and attacker's tendencies [23]. Such models can be used to assess the attractiveness of a target to an attacker if its apparent susceptibility is manipulated via its observables.

Axiom 1: Adding more vulnerabilities to one of the two equal-value systems increases the likeliness (until a threshold) of attack on the one with more vulnerabilities.

- *Dissuasion*: Dissuasion describes the steps taken by a defender to influence an attacker's behavior in mission's favor. It involves manipulating system observables to make it look like it has stronger security than it actually does. This is to discourage attackers from attacking it. As shown in Algo. 1, if the estimated resources for exploiting the system go over $\hat{r}$ or the estimated risk goes over $\hat{\rho}$, the attacker will be dissuaded from attacking the system. Dissuasion is generally implemented as deterrence or devaluation. Deterrence involves a false display of greater strength. Devaluation, on the other hand, involves manipulating observables to lessen the perceived value that comes out of compromising a system. McGill develops a probabilistic framework around the use of defensive dissuasion as a defensive measure [22]. Deception-based techniques are supplementary to conventional prevention techniques rather than a replacement.

Axiom 2: False display of strength dissuades an attacker from attacking the system.

Axiom 3: Increasing or decreasing the perceived value of a system affects the attacker's preference of attacking it favorably or adversely.

Honeypot is a tool of deception. It generally comes across as a system capable of a low-resource compromise with high perceived gains. Honeypot not only distracts an attacker from attacking the main system, but also heavily logs the attacker activities. Studying these logs can help the defender to gauge an attacker's capability and come up with a good strategy to ward off any future attacks. Spitzner describes honeypot as "a security deception resource whose value lies in being probed, attacked, or compromised" [40]. Honeypots are generally classified under two categories: Physical and Virtual honeypots. Physical honeypots are when real computer systems are used to create each honeypot. Virtual honeypots use software to emulate the workings of a real honeypot and the connecting network. They are cheaper to create and maintain and hence, are used in the production environments more often. Virtual honeypots are further divided into high-interaction and low-interaction honeypots. Qasswawi et al. provide a good overview of the deception-based techniques used in virtual honeypots [34].

High-interaction honeypots provide an emulation for a real operating system. Thus, the attacker can interact with the operating system and completely compromise the system. Some examples are User Mode Linux (UML), VMware, Argos, etc. Low-interaction honeypots simulate limited network services and vulnerabilities. They cannot be completely exploited. Examples are LaBrea, Honeyd, Nepenthes, etc. [33, 34].

Cohen's Deception Toolkit (DTK) laid the groundwork for low-interaction honeypots [5]. It led to the development of advanced products such as Honeyd [40]. Honeyd simulates services at TCP/IP level in order to deceive tools like Nmap and Xprobe. Though it does not emulate the entire operating system, its observables are modified to give the impression that it does.

Honeypot Farm is a cluster comprised of honeypots of the same or different kinds. Hybrid honeypot farms usually consist of a mixture of low and high-interaction honeypots.

5.2 Design

Building up from the concepts discussed above, we extend the model presented by Lakhani [17] to design our deception-based survivability framework.

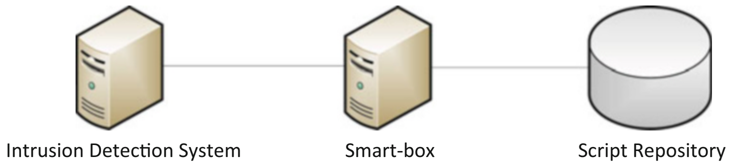


Fig. 1 Smart-box

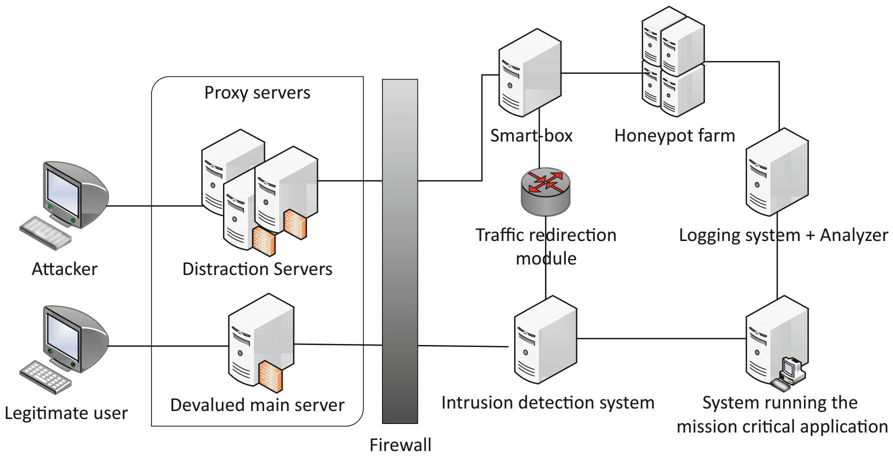

Proxy servers
Attacker
Distraction Servers
Legitimate user
Devalued main server
Firewall
Smart-box
Honeypot farm
Traffic redirection module
Logging system + Analyzer
Intrusion detection system
System running the mission critical application

Fig. 2 Deception framework for mission survivability

Smart-Box is a component that we use in the presented survivability framework to help figure out the best deception for suspected traffic flows. Conceptually, a smart-box works as shown in Fig. 1. It takes information from the intrusion detection system about the suspected traffic flow and figures out the AIOS based on it [19]. It, then, maps the AIOS to deception scripts that are stored in a script repository. These scripts are nothing but configuration files to choose and setup an appropriate honeypot in the honeyport farm.

Figure 2 shows the framework of a mission survivable (production) system that runs behind several layers of protection including firewalls, deception, etc. The first layer consists of proxy servers that use Axioms 1, 2, and 3 to attract attackers to choose vulnerable servers. Based on their choice of proxy servers, the traffic flows are marked suspicious or otherwise. The suspicious traffic is re-routed to the honeypot farm via a smart-box. Other traffic streams continue to go through the default proxy server, the firewall and the intrusion detection system. Intrusion detection system is another layer of defense which re-routes any suspicious traffic to the honeypot farm for further analysis.

Intrusion detection systems usually sieve out suspicious traffic based on two criteria: either the intrusion detection system identifies an attack pattern in the traffic flow or the traffic originates to/from dark address space. Dark address space is the set of Internet's routable addresses reserved for future network expansion. These two criteria worked just fine until cloud computing came along. Now attackers can launch their attacks from behind the cloud using valid IP addresses and evade detection. Therefore, in addition to the existing traffic-filtering methodologies, we introduce a layer of distraction proxy servers. This layer contains a main server which is widely publicized to legitimate clients. This main server is extremely secure and its observed security is further enhanced (deception/deterrence). Thus, amateur attackers are dissuaded from attacking it. Other proxy servers expose a specific set of non-essential, vulnerable services. For instance, one server can keep the *ssh* port open to accept the traffic, while the other can mislead the attacker into thinking that it is running a vulnerable version of Windows operating system. These servers not only distract the malicious traffic away from the main server but, also inform the smart-box about attackers' intentions (based on their preference of proxy servers and vulnerabilities that they try to exploit). Note that similar deception layers can be deployed throughout the system to capture and redirect malicious traffic.

Smart-box helps to identify the AIOS of a traffic flow. Additionally, it helps optimize resource allocation in hybrid honeypot farms. Honeypots should be assigned to each traffic flow based on its AIOS assessment. This is because low-interaction honeypots are lightweight but can be easily verified by an advanced attacker. Use of high-interaction honeypots is more fool-proof but consumes more computing and memory resources. Thus, smart-box helps in optimal assignment of the honeypots so more resources can be assigned to tackle advanced attacks.

Logging tools and analyzer in the honeypot farm work together to create a complete attack profile. Based on this attack profile, the traffic flow is either whitelisted and forwarded to the production server or blacklisted. If blacklisted, either automated patches, if available, are executed in the next recovery cycle, or a system administrator is alerted. This is the step where the attack profile helps the defender to develop an effective patch for the next recovery cycle, while the unsuspected malicious actor stays busy playing with the honeypot. Thus, deception buys defender the time to design an effective recovery. If imminent danger is not perceived, the defender can choose to wait through multiple recovery cycles until he is confident that the patch will be effective.

Since a system is "as secure as its weakest point", it is important to ensure that this survivability framework not only provides good security but is tamper-resistant at all times. Since all components in this design such as proxy servers, traffic redirection module, intrusion detection systems, etc. are connected to the same network, they are susceptible to intrusions. Therefore, these components need to be made tamper-resistant. Techniques, such as lightweight cyclic monitoring [24], can be employed to ensure that security-monitoring modules on all these components stay tamper-resistant. Integrity status of each component should be surreptitiously detected and reported to the production system for verification. This can be achieved by using the scheme described by Mehresh et al. in [26]. The detection is surreptitious so the attacker is not spooked and the production system is aware of which component to trust. This arrangement results in a cyclic integrity-check. All components make sure that production system works tamper-free at all times, while the production server takes care of the integrity-check of all components.

Anagnostakis et al. proposed shadow honeypots as an effective solution to deploy honeypots in a production environment [1]. Shadow honeypots use a combination of anomaly-based intrusion detection systems and shadow honeypots. A variety of anomaly detectors monitor traffic in the network and the suspicious traffic is forwarded to a shadow honeypot. Shadow honeypot is an identical copy of the production server but instrumented to detect potential attacks. Suspicious traffic is verified by the shadow and handled accordingly. We see many challenges in this approach. First, predictive anomaly detectors (higher sensitivity) will have more false positives and will direct more misclassified traffic to the shadow honeypot, creating unnecessary delays and overhead. Reactive anomaly detectors (lower sensitivity) will take more time to create a complete profile and may miss a lot of malicious traffic before identifying a problem in the flow. Moreover, identifying zero-day attacks ask for a higher sensitivity intrusion detection. Additionally, each suspicious traffic flow may need separate copies of shadow honeypot (else an attacker can verify deception by initiating two parallel malicious flows). This further increases the overhead.

6 Discussion and Conclusion

The most important factor to consider while designing a deception-based survivability framework is the fact that nothing can remain a secret if it is widely deployed. Hence, an effective deception must assume that an attacker knows about its existence with some probability. That's why all deceptions should be non-verifiable. For instance, in this case, when an attacker sees several proxy servers with vulnerabilities, he can send traffic flows to all the servers. The flow with the best response time is the one that does not go through the honeypot farm. That's why, any feedback loop for the users must also be controlled with deception. In this case, adding small random delays to the unsuspected traffic flows to ensure a uniform response time throughout. Discussing deception in feedback loop is beyond the scope of this chapter.

Another major challenge is the smart-box design. A smart-box performs two major functions: assess the nature of the traffic flow and, map the AIOS to a honeypot. Implementing both these functions is a major challenge and will benefit excessively from the use of machine learning algorithms. Deceptions in honeypots can also be made customizable based on the parameters provided by the smart-box. Other challenges such as designing proxy servers, re-routing, choosing the intrusion detection system, etc. are system and deception-specific.

This chapter identifies deception as the tool of defense against APT but also recognizes the moral, legal and other challenges that come along with it. The deception-based survivability framework presented is based on an underlying attack model that characterizes most APTs. Stealth and multi-phased approach are the two distinguishing features of APT. The presented framework deals with zero-day attacks while still conserving the timeliness property of a mission. It uses concepts of deception to introduce a layer of proxy servers that helps system to identify the suspicious traffic flows. Similar other deception-based layers can be deployed throughout the system to capture suspicious traffic flows. This traffic is then rerouted to a smart-box that identifies the AIOS associated with it. AIOS helps in the selection of an appropriate honeypot to which this traffic is further forwarded. Honeypots provide an important functionality of uncovering the stealthy patterns in these traffic flows with a higher probability in a shorter time. This way, the framework helps in identifying stealth attacks at an early stage.

A major advantage of this framework is the strong recovery that it provides. It buys defender more time to analyze the suspicious traffic flow without spooking the adversary. The analyzer and log modules help system administrators to design effective and targeted recovery procedures based on the good-quality attack profiles. This information can also help system administrators decide if they can wait through some recovery cycles to collect more information and improve the patches or deploy something right away (if system is in immediate danger). Hence, this framework ensures system survivability equipped with a strong recovery phase.

References

1. Anagnostakis, K.G., Sidiroglou, S., Akritidis, P., Xinidis, K., Markatos, E., Keromytis, A.D.: Detecting targeted attacks using shadow honeypots. In: Proceedings of the 14th Conference on USENIX Security Symposium, p. 9 (2005)
2. Bake, S., Filipiak, N., Timli, K.: In the dark: crucial industries confront cyberattacks. McAfee second annual critical infrastructure protection report (2011)
3. Baskerville, R.: Information warfare action plans for e-business. In: The 3rd European Conference on Information Warfare and Security, pp. 15–20 (2004)
4. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. (CSUR) **41**, 15:1–15:58 (2009)
5. Cohen, F.: Deception Toolkit (2001)
6. Cohen, F., Lambert, D., Preston, C., Berry, N., Stewart, C., Thomas, E.: A framework for deception. IFIP-TC11, Computers and Security (2001)
7. Cole, E.: Advanced Persistent Threat: Understanding the Danger and How to Protect your Organization. Syngress, Waltham (2012)
8. Daly, M.K.: The advanced persistent threat. In: Large Installation System Administration Conference (LISA) (2009)
9. Daniel, D.C., Herbig, K.L.: Strategic Military Deception. Pergamon Press, New York (1982)
10. Ellison, R.J., Fisher, D.A., Linger, R.C., Lipson, H.F., Longstaff, T.A., Mead, N.R.: Survivability: protecting your critical systems. IEEE Internet Comput. **3**, 55–63 (1999)
11. Falliere, N., Murchu, L.O., Chien, E.: W32. stuxnet dossier - White paper. Symantec Corporation, Security Response (2011)
12. Goyal, R., Sharma, S., Bevinakoppa, S., Watters, P.: Obfuscation of stuxnet and flame malware. Latest Trends in Applied Informatics and Computing (2012)
13. Gross, M.J.: A Declaration of Cyber-War (2011)
14. Kapoor, A., Mathur, R.: Predicting the future of stealth attacks. Virus Bulletin Conference (2011)
15. Kartaltepe, E.J., Morales, J.A., Xu, S., Sandhu, R.: Social network-based botnet command-and-control: emerging threats and countermeasures. Proceedings of the 8th International Conference on Applied Cryptography and Network Security (ACNS), pp. 511–528 (2010)

16. Knappa, K.J., Boulton., W.R.: Cyber-Warfare Threatens Corporations: Expansion into Commercial Environments. Inf. Syst. Manag. **23**, 76–87 (2006)
17. Lakhani, A.D.: Deception techniques using Honeypots. MSc Thesis, University of London, ISG, Royal Holloway. (2003)
18. Levine, J.G., Grizzard, J.B., Owen, H.L.: Using honeynets to protect large enterprise networks. IEEE Secur. Priv. **2**, 73–75 (2004)
19. Liu, P., Zang, W., Yu, M.: Incentive-based modeling and inference of attacker intent, objectives, and strategies. ACM Transactions on Information and System Security (TISSEC), vol. 8 (2005)
20. Masood, R., Um-e-Ghazia, U., Anwar, Z.: SWAM: stuxnet worm analysis in metasploit. Frontiers of Information Technology (FIT), pp. 142–147 (2011)
21. McAfee Labs and McAfee Foundstone Professional Services: Protecting your critical assets, lessons learned from Operation Aurora. Technical Report (2010)
22. McGill, W.L.: Defensive dissuasion in security risk management. In: IEEE International Conference on Systems, Man and Cybernetics (SMC) (2009)
23. McGill, W.L., Ayyub, B.M., Kaminskiy, M.: Risk analysis for critical asset protection. Risk Anal. **27**(5), 1265–1281 (2007)
24. Mehresh, R., Rao, J.J., Upadhyaya, S.J., Natarajan, S., Kwiat, K.: Tamper-resistant monitoring for securing multi-core environments. In: International Conference on Security and Management (SAM) (2011)
25. Mehresh, R., Upadhyaya, S.J.: A deception framework for survivability against next generation cyber attacks. In: International Conference on Security and Management (SAM) (2012)
26. Mehresh, R., Upadhyaya, S.J., Kwiat, K.: Secure proactive recovery - a hardware based mission assurance scheme. J. Netw. Forensics **3**, 32–48 (2011)
27. Munro, K.: Deconstructing flame: the limitations of traditional defences. Computer Fraud and Security, pp. 8–11 (2012)
28. Murphy, B.S.: Deceiving adversary network scanning efforts using host-based deception. Technical Report, Air Force Institute of Technology, Wright-Patterson Air Force Base (2009)
29. Nakashima, E., Pomfret, J.: China proves to be an aggressive foe in cyberspace (2009)
30. Neagoe, V., Bishop, M.: Inconsistency in deception for defense. In: Proceedings of the 2006 workshop on New security paradigms (2007)
31. Patel, R.R., Thaker, C.S.: Zero-day attack signatures detection using honeypot. International Conference on Computer Communication and Networks (CSI- COMNET) (2011)
32. Portokalidis, G., Bos, H.: SweetBait: zero-hour worm detection and containment using low- and high-interaction honeypots. Sci. Direct **51**, 1256–1274 (2007)
33. Provos, N., Holz, T.: Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison-Wesley, Boston (2008)
34. Qassrawi, M.T., Zhang, H.: Deception methodology in virtual honeypots. In: Second International Conference on Networks Security Wireless Communications and Trusted Computing (NSWCTC), vol. 2, pp. 462–467, 24–25 (2010)
35. Ramilli, M., Bishop, M.: Multi-stage delivery of malware. In: Proceedings of the 5th International Conference on Malicious and Unwanted Software (MALWARE) (2010)
36. Repik, K.A.: Defeating adversary network intelligence efforts with active cyber defense techniques. Master's Thesis, Graduate School of Engineering and Management, Air Force Institute of Technology (2008)
37. Rowe, N.C., Rothstein, H.S.: Two taxononmies of deception for attacks on information systems. J. Inf. Warfare **3**, 27–39 (2004)
38. Smart, M., Malan, G.R., Jahanian, F.: Defeating TCP/IP stack fingerprinting. In: Proceedings of the 9th Conference on USENIX Security Symposium, vol. 9, pp. 17–17 (2000)
39. Smith, A., Toppel, N.: Case study: using security awareness to combat the advanced persistent threat. In: Thirteenth Colloquium for Information Systems Security Education (2009)
40. Spitzner, L.: Honeynet Project, Know Your Enemy: Defining Virtual Honey-nets (2008)

41. Tankard, C.: Advanced persistent threats and how to monitor and deter them. Netw. Secur. **2011**, 16–19 (2011)
42. Tzu, S.: The Art of War (Translated by James Clavell). Dell Publishing, New York (1983)
43. Watson, D., Smart, M., Malan, G.R., Jahanian, F.: Protocol scrubbing: network security through transparent flow modification. IEEE/ACM Trans. Networking **12**, 261–273 (2004)
44. Yuill, J., Denning, D., Feer, F.: Using deception to hide things from hackers: processes, principles, and techniques. J. Inf. Warfare **5**, 26–40 (2006)