Implementation Attacks: From Theory to Practice

Dissertation

---

zur Erlangung des Grades eines Doktor-Ingenieurs
der Fakultät für Elektrotechnik und Informationstechnik
an der Ruhr-Universität Bochum

---

by David Oswald
Bochum, September 2013

To my sunce ljubavi for her love and support

David Oswald
Place of birth: Essen, Germany
Author's contact information:
david.oswald@rub.de
www.kasper-oswald.de

Thesis Advisor:	**Prof. Dr.-Ing. Christof Paar**
	Ruhr-Universität Bochum, Germany
Secondary Referee:	**Prof. David Naccache**
	Université Paris II / ENS, France
Thesis submitted:	June 18, 2013
Thesis defense:	August 1, 2013.
Last revision:	September 16, 2013.

Du bist wie eine Blume,
So hold und schön und rein;
Ich schau dich an, und Wehmut
Schleicht mir ins Herz hinein.

Mir ist, als ob ich die Hände
Aufs Haupt dir legen sollt,
Betend, daß Gott dich erhalte
So rein und schön und hold

Heinrich Heine.

Abstract

Embedded devices have become omnipresent both in business applications and in the private sector. Various, highly diverse types of systems rely on embedded computing devices that, compared to traditional PCs, are often not directly noticed by the user. Examples for such embedded devices are manifold: RFID-based smartcards are on the way to replacing magnetic-stripe cards for payments and are wide-spread in public transport and user identification. Long-range RFID transponders are used for identifying and managing large assets like freight containers in the supply chain. A modern car comprises a large number of microcontrollers, controlling numerous components ranging from the engine to the immobilizer. Consumer products like mobile phone batteries and inkjet cartridges are equipped with special-purpose chips to prevent counterfeit. Access to online services is secured with hardware tokens for two-factor authentication, and electronic door locks for controlling the physical access to buildings and offices are on their way to replacing traditional mechanical systems.

With the increasing reliance on embedded devices arises the need to thoroughly examine the related security mechanisms. Malicious adversaries can cause severe financial losses, e.g., due to IP counterfeit, industrial espionage, or fraudulent payment transactions, but also endanger material and even human lives, e.g., in the case of vehicles, medical devices, or industrial machinery. Hence, many embedded devices incorporate protection mechanisms, often involving cryptographic functionality to fulfill requirements like confidentiality, authenticity, and integrity. To evaluate the security level of according implementations, a purely mathematical point of view is not sufficient:

Classical cryptanalysis considers an algorithm as an ideal construct for which only the inputs and outputs are known to an adversary, however, for real-world devices, this assumption does not hold. The actual hardware or software realization of a cryptographic primitive is subject to *implementation attacks* that utilize physical properties to break analytically secure schemes like AES, 3DES, or RSA. Active *fault injection* techniques allow to extract secret information by disturbing the cryptographic computation, e.g., exposing the device to over and undervoltages or UV-C and laser light. In contrast, passive *side-channel analysis* is based on monitoring the behavior of the target device while executing a cryptographic algorithm, e.g., by measuring the power consumption or the EM emanation.

Interestingly, the practical threat posed by implementation attacks and particularly side-channel analysis to embedded devices has received relatively little attention in the scientific community. This is in contrast to the in-depth treatment of the theory of such attacks in the scientific community. The goal of this thesis is to fill this gap and evaluate the physical security of widely used devices, with a special focus on side-channel analysis. We propose improved digital and analog pre-processing techniques to increase the quality of side-channel measurements and describe tools to perform implementation attacks at a low cost. Amongst others, we present circuits to isolate the leakage of contactless smartcards and introduce the

GIAnT, a cost-efficient FPGA-based platform for controlling target devices and conducting side-channel and fault injection attacks.

Applying the developed tools, the cryptographic functionality of several products representing different classes of embedded devices is analyzed. The examples include RFID transponders used in the supply chain, a contactless smartcard (the Mifare DESFire MF3ICD40), an electronic locking system by SimonsVoss, a two-factor authentication token (the Yubikey 2), SHA-1-based anti-counterfeiting chips by Maxim Integrated, and Altera Stratix II FPGAs. In all cases, we show that implementation attacks and side-channel analysis in particular can circumvent the (cryptographic) protection of the examined devices. Since the demonstrated attacks can be put into practice with a relatively low financial effort, it appears to be mandatory to include suitable countermeasures in new product generations. In this regard, we discuss possible approaches, taking the often limited resources available in embedded applications into account. As a concluding result, we outline directions for further research, both from an adversary's point of view and from a constructive perspective.

Keywords.

Cryptography, implementation attacks, side-channel analysis, real-world attacks, contactless smartcards, Mifare DESFire MF3ICD40, RFID, electronic locks, SimonsVoss, two-factor authentication, Yubikey, anti-counterfeit, Altera Stratix II, bitstream encryption, reverse-engineering, IP protection, countermeasures

Kurzfassung

Implementierungsangriffe: Von der Theorie zur Praxis

Eingebettete Systeme sind heute sowohl im geschäftlichen Bereich als auch in Produkten für Endkunden allgegenwärtig. Eine Vielzahl stark unterschiedlicher Anwendungen basiert auf eingebetteten Prozessoren, die verglichen mit traditionellen PCs für den Anwender oft unsichtbar sind. Die Beispiele in dieser Hinsicht sind zahlreich: Kontaktlose Smartcards ersetzen klassische Karten mit Magnetstreifen in Bezahlsystemen und sind schon heute weit verbreitet im öffentlichen Nahverkehr und für Identitätsdokumente. RFID-Transponder mit hohen Reichweiten werden zur Identifikation von z.B. Frachtcontainern in der Logistik eingesetzt. Ein modernes Auto beinhaltet viele Mikrocontroller, die Steuerungsaufgaben vom Motor bis hin zur Wegfahrsperre übernehmen. Massenprodukte wie Mobiltelefon-Akkus oder Druckerpatronen werden mit speziellen Chips versehen, um Produktpiraterie entgegenzuwirken. Der Zugang zu Online-Anwendungen wird mit Zwei-Faktor-Authentifizierung auf Basis von Hardware-Tokens gesichert und elektronische Schließsysteme ersetzen mechanische Lösungen.

Im Zuge der stetig wachsenden Verbreitung eingebetteter Systeme stellt sich die Frage nach der Sicherheit in entsprechenden Anwendungen. Böswillige Angreifer können neben finanziellen Verlusten—z.B. durch Fälschung von Produkten, Industriespionage oder betrügerische Zahlungs-Transaktionen—auch unmittelbar Sachschäden herbeiführen und im schlimmsten Fall Menschenleben gefährden, z.B. bei Automobilen, medizinischen Geräten oder Industrieanlagen. Daher verfügen zahlreiche eingebettete Geräte über entsprechende Schutzmaßnahmen. Oft wird Kryptographie verwendet, um Sicherheitsziele wie Vertraulichkeit, Authentizität und Integrität zu realisieren. Um das tatsächliche Sicherheitsniveau entsprechender Implementierungen zu bewerten, ist im Kontext eingebetteter Systeme eine rein mathematische Untersuchung nicht hinreichend:

Die klassische Kryptanalyse betrachtet einen kryptographischen Algorithmus als ein idealisiertes Konstrukt, bei dem für einen Angreifer nur die Ein- und Ausgabewerte bekannt sind. Für praktische Umsetzungen ist diese Annahme jedoch ungültig. *Implementierungsangriffe* zielen unmittelbar auf die Hard- und Software eines Gerätes und verwenden physikalische Eigenschaften, um mathematisch sichere Verfahren wie AES, 3DES oder RSA zu brechen. So kann z.B. mittels aktiver *Fehlerinjektion* die Ausführung eines Algorithmus gestört und geheime Daten extrahiert werden. Im Gegensatz dazu basieren *Seitenkanal-Analysen* auf der passiven Messung physikalischer Größen, z.B. der Stromaufnahme oder der EM-Abstrahlung einer Schaltung.

Die praktische Bedrohung durch Implementierungsangriffe und insbesondere durch Seitenkanal-Analyse wurde bislang in geringem Maße wissenschaftlich untersucht. Das Ziel dieser Dissertation ist es daher, in dieser Hinsicht weitere Untersuchungen an weitverbreiteten eingebetteten Systemen mit einem speziellen Fokus auf Seitenkanal-Analysen durchzuführen. Zu diesem Zweck werden zunächst analoge wie digitale Vorverarbeitungsmethoden betrachtet, die die Qualität entsprechender Messdaten erhöhen können. Darüber hinaus werden Werkzeuge vorge-

stellt, mit denen Implementierungsangriffe kostengünstig umgesetzt werden können. Dies beinhaltet u.a. Schaltungen zur Rückgewinnung des Seitenkanal-Signals kontaktloser Smartcards und GIAnT, eine FPGA-Plattform zur Steuerung der analysierten Geräte und zur Realisierung von Seitenkanal- und Fehlerinjektions-Angriffen.

Mit den entwickelten Werkzeugen wird im Anschluss die Sicherheit spezifischer Produkte—die repräsentativ für verschiedene Arten von eingebetteten System sind—geprüft. Im Rahmen der durchgeführten Fallstudien werden RFID-Transponder für Supply Chain Anwendungen, eine kontaktlose Smartcard (Mifare DESFire MF3ICD40), ein elektronisches Schließssystem von SimonsVoss, ein Zwei-Faktor Authentifizierungstoken (Yubikey 2), SHA-1-basierte Chips zum Fälschungsschutz von Maxim Integrated und Altera Stratix II FPGAs untersucht. In allen Fällen zeigte sich, dass mit Implementierungsangriffen und insbesondere Seitenkanal-Analysen die (kryptographischen) Schutzmaßnahmen erfolgreich umgangen werden können. Da die vorgestellten Angriffe mit relativ kostengünstiger technischer Ausrüstung umgesetzt wurden, erscheint es unumgänglich, bei der Entwicklung neuer Produkte entsprechende Gegenmaßnahmen zu berücksichtigen. Mögliche Ansätze in dieser Hinsicht werden unter Berücksichtigung der oft begrenzten Ressourcen in eingebetteten Anwendungen betrachtet. Abschließend werden relevante Forschungsthemen im Bereich der eingebetteten Sicherheit—sowohl aus der Sicht eines Angreifers als auch eines Entwicklers—identifiziert und näher beschrieben.

Schlagworte.

Kryptographie, Implementierungsangriffe, Seitenkanal-Analyse, Praktische Angriffe, Kontaktlose Smartcards, Mifare DESFire MF3ICD40, RFID, Elektronische Schließsysteme, SimonsVoss, Zwei-Faktor Authentifizierung, Yubikey, Schutz vor Produktfälschungen, Altera Stratix II, Bitstrom-Verschlüsselung, Reverse-Engineering, Gegenmaßnahmen

Acknowledgements

It has been about four years since I began working on my PhD at the Chair for Embedded Security, and now finally I managed to write all the stuff down. So it is only fitting—before starting with the dull science—to say thanks to the people that enabled me to go my way.

First of all, I want to send all my love to the wonderful Jelena Ninić who shines like an always bright sun in my live—and who drew the fantastic below picture before my defense to help me relax. Sunce, thanks for all your support and love, volim te!

Lots of thanks go to my parents Hetti and Detlef for giving me the opportunity to study in the first place and teaching me many important things about life and science. Thanks also to the rest of my vast family and all my friends, you are great, guys!

Coming to the professional part, first, I'd like to thank my advisor Christof Paar who has been an excellent teacher (and not only for scientific matters) in the past years. With his broad perspective and open-mindedness, Christof has a unique way of giving interesting topics to the right people. 1000 Dank! Thanks go also to David Naccache for co-advising my thesis and managing to combine this job with his numerous other duties and responsibilities—merci!

Next in line are all the people who I got to know during my time at Emsec. First, thanks go to Timo Kasper, with whom I share a (chaotic) office (and phone) for many years now, wrote many papers and went to many conferences with, and with whom I finally managed to set up a company. Danke, Dr. Kasper. Besides, I am obliged to the non-scientific staff at Emsec, that is, Irmgard (our team assistant) and Horst (our admin). Thanks for keeping the group up and running. Of course, I also did not forget my various co-authors, so here is the list (in alphabetical order): Roberto Avanzi, Benedikt Driessen, Tim Güneysu, Ilya Kizhvatov, Alexander Kühn, Gregor Leander, Nilesh Madhu, Ingo von Maurich, Amir Moradi, Bastian Richter, Falk Schellenberg, Jörn-Marc Schmidt, Daehyun Strobel, Pawel Swierczynski, Michael Tunstall, and Christian Zenger. Thanks, guys!

Finally, when thinking about whom to thank, some other people I owe gratitude to came to my mind: The students who I supervised and who helped me to advance with my PhD, the guys at CRI (where I was intern for a few months), Thomas Eisenbarth (and Ricarda and Finn and David) for their hospitality, the people at my sports club (Essener Hap Ki Do Sportclub e.V.), and a few university teachers who I particularly remember: Roberto Avanzi, Helmut Jacob, and Günter Felbecker. I hope I did not forget anyone, but if I did and I owe you thanks, you probably know. Having said that, let us come to the science, then—and to motivate what follows with the words of Carl Sagan: "We live in a society exquisitely dependent on science and technology, in which hardly anyone knows anything about science and technology."

Thanks!

Table of Contents

Part I

Preliminaries

Chapter 1

Introduction

In this chapter, the motivation for the research described in the thesis is established. Presenting numerous security-relevant applications of embedded devices, we introduce implementation attacks and show the need for a thorough analysis of real-world cryptographic devices with respect to this attack vector. Finally, we outline the structure of this thesis and summarize the main research contributions that are described in detail in the subsequent chapters.

Contents of this Chapter

1.1 Motivation

Following the prevalent use of PCs in business and private applications, embedded computing devices have also become wide-spread in the past years. Numerous systems rely on Microcontrollers (µCs), Field Programmable Gate Arrays (FPGAs), or Application Specific Integrated Circuits (ASICs). These computing devices are often invisible to the end-user. For example, a modern low-end car contains approximately 30 to 50 separate electronic units [iee09], controlling, amongst others, the engine, antilock brakes, the immobilizer, and the door locks. In contrast to a PC, an embedded system is usually not directly identifiable as such by a user and at the same time tends to be more safety or security-relevant:

For a car, an industrial machine, or an airplane, accidental malfunctions of the electronics or manipulations intentionally performed by an adversary can lead to severe material and monetary damage and in the worst case endanger human lives. In the context of electronic access control, e.g., wireless-controlled door locks, flaws in the underlying embedded system can enable an attacker to gain unauthorized access to a building, a threat especially relevant in the context of industrial espionage. Radio Frequency IDentification (RFID) transponders employed in contactless payment and ticketing systems may—if not sufficiently secured—be duplicated, incurring financial losses for the system operator [KSP10].

A different threat for RFIDs and Radio Frequency (RF) devices in general originates from the wireless nature of the interface. An adversary can both passively eavesdrop on legitimate

communication or actively query an RFID transponder. Since the access to the wireless link cannot be easily restricted (as, e.g., for a cable medium), corresponding attacks can for instance affect the privacy of the users or allow an adversary to locate assets like freight containers in a supply chain RFID system.

In the context of industrial espionage, attacks on embedded devices can be used to prepare the actual theft of confidential information. For example, corporate networks are today often secured with two-factor authentication, involving a hardware token in addition to the traditional username-password credentials. This rules out attacks that test passwords from a dictionary or perform an exhaustive search. However, as a consequence, the second authentication factor becomes subject to attacks itself. This was the case in the example of the attack on RSA's SecurID system [Bri11]. Although in this specific situation weaknesses of the backend network had been exploited, the attackers would certainly also have utilized flaws of the hardware token if they would have found any.

A 2011 study [Det11] regarding the costs due to cyber crime in the UK confirms the severity of this problem: While the cost to private users and governmental institutions is estimated to be £ 3.1 billion and £ 2.2 billion per year, respectively, [Det11] puts the figure for UK businesses at £ 21 billion per year. Although the magnitude of these numbers was questioned, e.g., in [Bre11], the report certainly allows to draw qualitative conclusions. In particular, it is interesting to examine which kind of attacks lead to the cost of £ 21 billion. Figure 1.1 summarizes the results of [Det11] in this respect, with Intellectual Property (IP) theft and industrial espionage accounting for approximately 79% of the overall financial damage. Surprisingly, more "obvious" attacks like extortion, direct online theft, and the loss of customer data are less dominant with a share of only 21%.

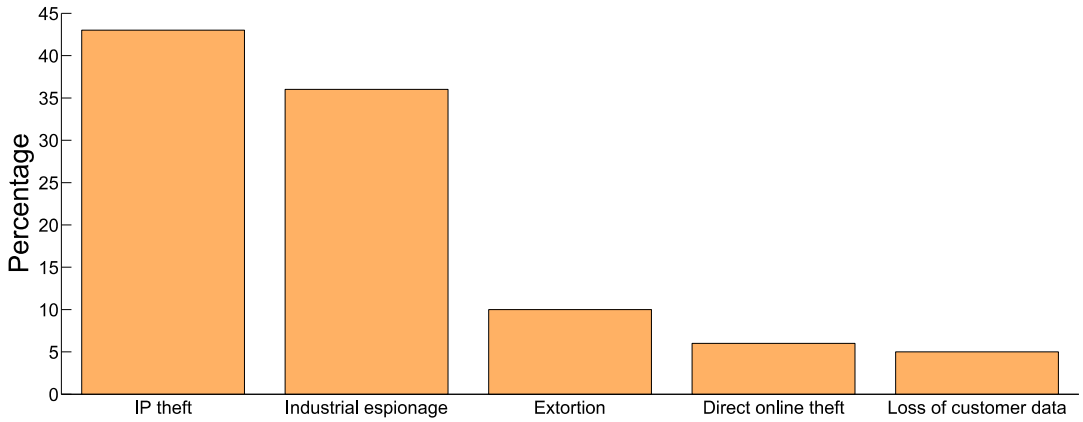


Figure 1.1: Distribution of costs for UK businesses due to cyber crime according to [Det11]

Note that IP theft ranks highest in Fig. 1.1 with over 40% of the total cost. Although IP theft is today mostly performed via the Internet, the high cost underlines the general relevance of the problem. Thus, from the perspective of the manufacturer of an (electronic) product, security may also mean that the product cannot be counterfeit by a competitor. This is the case both for low-end mass products like cellphone batteries or inkjet cartridges, but may also apply to expensive equipment like high-end network routers. Thus, corresponding devices either

use hardware components that inherently comprise protection mechanisms against counterfeit or contain additional special-purpose Integrated Circuits (ICs). In this case, an adversary might go to great lengths to overcome these countermeasures, since one-time access to the IP of one specific device allows to build duplicates of the respective product.

Certainly, there are numerous other kinds of applications that employ embedded systems with security-relevant functionality. However, in this thesis, we focus on case studies representing the mentioned examples, i.e., RFIDs, electronic locks, two-factor authentication tokens, anti-counterfeit ICs, and FPGAs.

As our targets or Devices Under Test (DUTs), we selected devices that are *(i)* in wide-spread use and *(ii)* were considered secure so far. Note that our research is not purely "destructive". Instead, we believe that an analysis of real-world products, eventually leading to successful attacks, is an important prerequisite for designing more secure embedded systems—which may, as outlined above, be used in cases where a failure can have severe consequences.

In a certain sense, the current state of the security of embedded devices can be compared to the situation for software when PCs became increasingly wide-spread. Until suitable attack techniques had been developed and been applied to real programs, most software (including operating systems) was highly susceptible to many kinds of malicious attacks. Only the cooperation between well-meaning "white hat hackers" and software developers helped to partially solve this problem—and even today, although many countermeasures are available, software is often still vulnerable due to its high complexity.

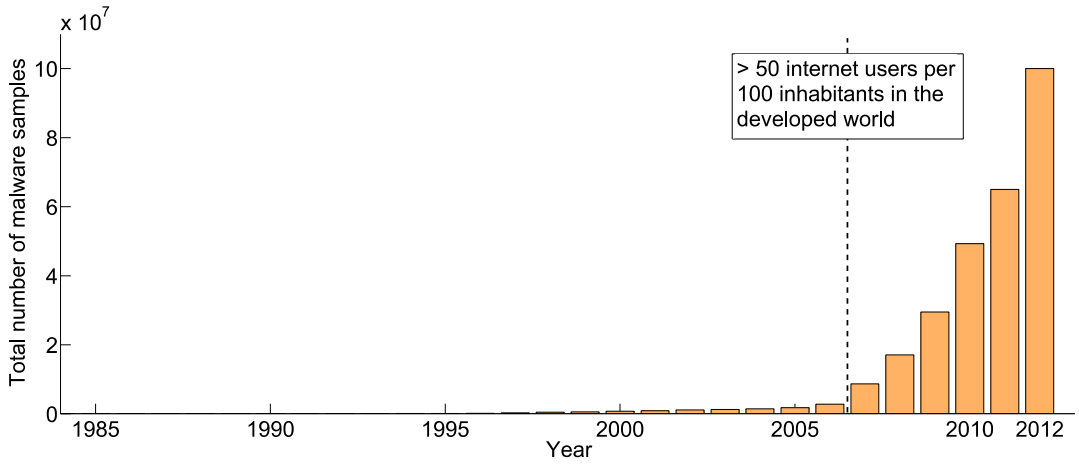


Figure 1.2: Total number of malware samples in the AV-Test database from 1984–2012 [AV-13]

In the case of attacks on software, a particular kind of organized crime has developed, focusing on exploiting flaws to, e.g., obtain login credentials and payment-related information or to create "botnets" used for illegal purposes. Two main reasons are in our eyes responsible for this situation: First, with most computers being connected via the Internet, newly discovered attacks can be easily carried out and quickly affect a huge number of PCs. As evident in Fig. 1.2, the number of PC malware samples in the AV-Test database [AV-13] began to strongly increase around 2006/2007. In these years, for the first time more than 50% of the inhabitants of developed countries had access to the Internet [Int11]. Second, until large-scale attacks started to appear on a regular basis, there was no obvious incentive for software manufacturers to

invest in security. Hence, the "investments" for criminal businesses to find and exploit flaws in wide-spread software packages were initially rather low.

To avoid a similar situation for embedded systems and to make sure that criminals are not the first ones to discover and exploit weaknesses, extensive research has to be carried out in this regard. To this end, in this thesis, we study several embedded systems and report the discovered security problems. Note that in most cases, the necessary equipment has a cost of less than USD 5,000, that is, the often hardware-based attacks could be performed at low cost without a complex and expensive lab setup.

Note that we aim at working with vendors to improve their products. Thus, as part of a responsible disclosure process, when we had found weaknesses, we informed the respective manufacturer several months before publishing our results. Depending on the concrete case and the resulting threat for practical applications, at times we left out certain details in our publications to prevent misuse of our findings. Also, we discussed suitable changes to a product with the vendor in order to help enhance the level of security.

1.2 Implementation Attacks

The field of *cryptology* can be subdivided into several areas. Following the classification of [PP10], *cryptography* deals with the design and implementation of cryptographic primitives, e.g., ciphers, hash functions, or signature schemes, and protocols which use these building blocks, e.g., for achieving confidentiality, integrity, or authentication. On the other hand, *cryptanalysis* attempts to find flaws in cryptographic constructions in order to estimate the actual security level of a scheme. In a continuous process, cryptanalysts analyze and potentially break cryptographic algorithms to allow designers to build systems not susceptible to the previously discovered attacks. In the long term, this led to the development of thoroughly analyzed and presumably highly secure algorithms like the AES.

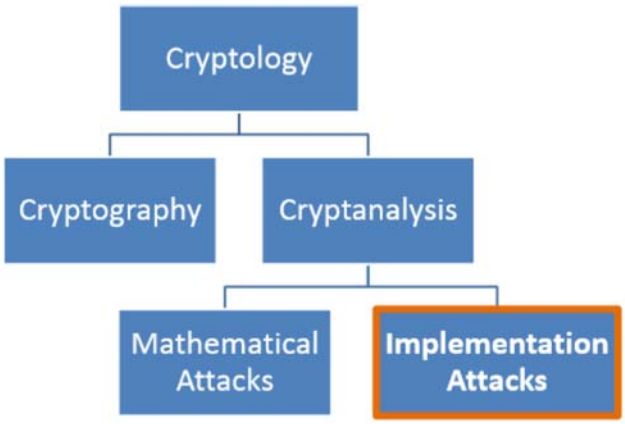


Figure 1.3: Areas of cryptology according to [PP10]

Implementation attacks form a special class of cryptanalytical techniques. Traditional, mathematical analysis treats a cryptographic algorithm as an ideal, atomic operation, i.e., all com-

putations are performed correctly and internal states are never available to an adversary. While this assumption may (up to a certain degree) hold for PCs exchanging messages over a network, it is violated when an adversary has physical access to the actual implementation of cryptography, e.g., on a µC, FPGA, or ASIC.

In this case, several new ways for attacking the implementation rather than the underlying mathematical theory open up. For example, an adversary may disturb the computations on a device and subsequently evaluate the "faulty" results. This method termed Fault Injection (FI) was first described in [BDL97] and has since then been adapted to most cryptographic algorithms, including otherwise secure primitives like AES, 3DES, RSA, and Elliptic Curve Cryptography (ECC).

Shortly after the discovery of FI, Kocher et al. proposed Side-Channel Analysis (SCA) in 1999 [KJJ99]. In contrast to the active manipulation of a device required for FI, SCA is based on the passive observation of the target device, e.g., measuring the power consumption or Electro-Magnetic (EM) emanation while the cryptographic algorithm is executed. Using statistical methods, secret information, e.g., a cryptographic key, can be extracted in this way, often without tampering with the device. An overview over these and additional implementation attacks is given in Fig. 1.4.

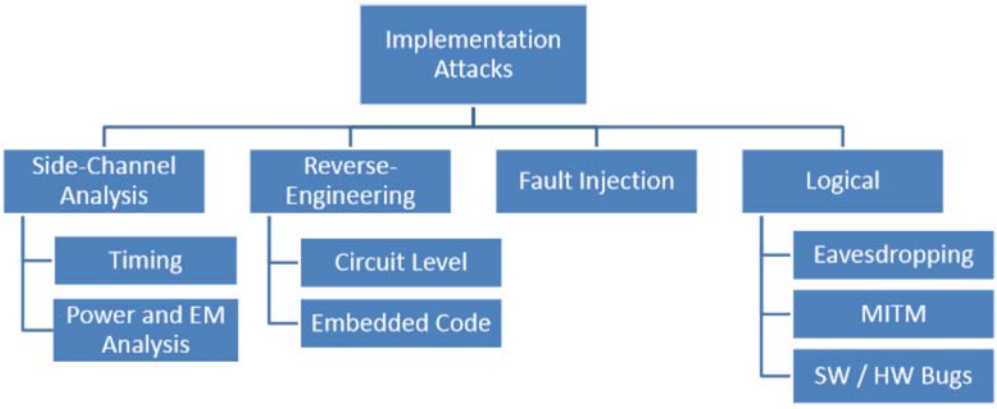


Figure 1.4: Types of implementation and implementation-related attacks

Reverse-engineering can be seen as an implementation-related attack in the context of embedded systems: given access to embedded program code or an implementation in silicon on an IC, an adversary attempts to reconstruct for instance confidential, vendor-specific algorithms. In certain cases, reverse-engineering can serve as a preparatory step for side-channel attacks or FI to simplify the profiling of a DUT or to be able to communicate with the device at all (in the case of proprietary protocols). *Logical attacks* focus on flaws on the level of transmission and communication protocols. For example, an adversary may eavesdrop on unprotected data exchanges or act as a Man-In-The-Middle (MITM) between legitimate communication partners. Finally, embedded software and hardware implementations can, like every complex system, contain bugs that allow an attacker to compromise the security by, e.g., sending unexpected input data to a DUT.

Implementation attacks are techniques that initially originated from the practical analysis of real devices. For example, at the time of the discovery of SCA, high-security smartcards were allegedly susceptible to this attack vector. However, since vendors reacted relatively quickly and included countermeasures to prevent straightforward implementation attacks, the focus of the respective research area shifted to a more theoretical perspective, e.g., studying the statistical aspects of SCA.

Until 2008, there were no further reports of SCA being successfully applied on a large scale in the real world. The break of the wide-spread KeeLoq Remote Keyless Entry (RKE) scheme [EKM+08] using a side-channel attack changed this situation. Since SCA could in this case be used to extract a master key that allowed an adversary to compromise all products of a specific vendor, the practical threat due to implementation attacks became evident. In this thesis, we continue the analysis of real-world products, applying state-of-the-art SCA and, in addition, occasionally also related techniques like FI or reverse-engineering. Taking representative examples for different kinds of embedded systems as outlined in Sect. 1.1, we demonstrate the potential consequences of insufficient protection against implementation attacks.

1.3 Structure of this Thesis

The steps performed for achieving the goal of this thesis can be summarized as follows:

Fundamentals In Chap. 2, we recapitulate the current state-of-the-art for SCA, focusing on methods that are generally applicable. In the context of pre-processing of side-channel measurements, we develop a technique to partially automate the choice of respective parameters.

Tools As a prerequisite for the practical application of implementation attacks, suitable tools for the communication with target devices, the acquisition and filtering of side-channel measurements, and for FI are necessary. In Chap. 3–5, we introduce the low-cost equipment utilized for this purpose.

Application Using the described SCA methods and the developed tools, we analyze several real-world devices and demonstrate the numerous security problems we came across. The selected target devices are examples for the following embedded systems:

(1) RFID transponders (Chap. 6 and Chap 7),

(2) Electronic locking systems (Chap. 8),

(3) Two-factor authentication tokens (Chap. 9),

(4) Anti-counterfeit ICs (Chap. 10), and

(5) FPGAs (Chap. 11).

Countermeasures Examining the results of the practical analyses from a more general perspective, we identify reasons for the discovered security problems in Chap. 12. On this basis, suitable countermeasures on different levels are described.

Future Work To conclude this thesis, in Chap. 13 directions for further research are outlined, both from the "destructive" perspective of the cryptanalyst and the constructive designer's point of view.

In the following Sect. 1.4, we give further details on the respective research contributions and the publications that form the basis for the results reported in this thesis.

1.4 Summary of Research Contributions

As mentioned, the research described in this thesis was conducted in several related areas. The contributions include improved processing techniques for SCA and the development of a cost-effective toolchain for implementation attacks. As the main result, several DUTs are analyzed with respect to the susceptibility towards implementation attacks.

1.4.1 SCA Pre-Processing

In the context of SCA, the measured side-channel signals are often subject to pre-processing before carrying out the actual attack. The parameters for the pre-processing, e.g., the cutoff frequency of a digital filter, are as of today mostly chosen manually. In Sect. 2.5, a method for the more systematic derivation of parameters for linear filters is described. Note that the proposed approach was developed following the practical attacks of Chap. 7–11, for which we often determined suitable pre-processing steps by exhaustively testing many parameter choices. *The research in this section was published in [OP13].*

1.4.2 Tools

The toolchain used for this thesis comprises numerous modules due to the manifold tasks one faces when analyzing real products. Apart from software components for data acquisition, processing, and statistical evaluation, oscilloscopes and an FPGA-based setup for implementation attacks were utilized. Custom analog filters were developed to increase the success rate of SCA of contactless RFID smartcards.

Measurement and Evaluation

For acquiring side-channel signals, two Digital Storage Oscilloscopes (DSOs) were employed to digitize measurements of the power consumption and the EM emanation of a DUT, cf. Sect. 3.2. For controlling the measurement process and for evaluating the obtained waveforms, a flexible framework for according software applications was developed, cf. Sect. 3.2.3 and Sect. 3.3. In addition to tools specifically designed for side-channel attacks, we also employed Software-Defined Radio (SDR) devices for receiving and transmitting RF signals as described in Sect. 3.1. The SDRs were, amongst others, utilized for determining the actual ranges of RFIDs in Chap. 6.

Pre-Processing for SCA of RFID

When performing SCA of contactless smartcards, the presence of a strong reader signal complicates the direct measurement of the side-channel leakage of the DUT. Special analog circuits are beneficial to isolate and amplify the relevant signal before converting it into digital data with a DSO. In Chap. 4, such components based on incoherent demodulation are introduced. The developed demodulator is subsequently applied for the analysis of the Mifare DESFire MF3ICD40 smartcard in Chap. 7. *The research in this section was published in [KOP12].*

GIAnT: A Platform for Implementation Attacks

Since commercial solutions for conducting implementation attacks are usually expensive, we created the custom, open-source platform GIAnT presented in Chap. 5. The device can for instance be employed to inject faults by means of supply voltage manipulations, to digitize analog signals, and to communicate with a wide variety of potential DUTs. Verifying the effectiveness of our platform, we show that cryptographic implementations on (unprotected) Atmel ATMega and ATXMega μCs can be attacked by means of voltage manipulations. Although the specifications of the GIAnT cannot compete with those of expensive high-end equipment, the provided functionality was sufficient for the purposes of this thesis. Moreover, because the GIAnT can be built for approximately USD 300, basic techniques for implementation attacks are made accessible to a wide audience, e.g., allowing designers to directly test their products. *This contribution is an improved version of the author's research published in [KOP10].*

1.4.3 Practical Implementation Attacks

Applying the developed toolchain, we analyze embedded cryptographic devices with respect to implementation attacks and SCA in particular. Doing so, we found weaknesses in several wide-spread commercial products, allowing to extract secrets like cryptographic keys within a few minutes to a few hours. Note that in the case of a successful attack, the affected vendor was informed several months ahead of time before the publication. We discussed suitable countermeasures and provided a summary of our results to allow the manufacturer to inform the customers before our discovery was made public.

RFID Range Measurements

As an introductory example, we demonstrate the problems that arise when protocols lacking cryptographic protection are executed over wireless links in Chap. 6. More precisely, we show that the effective range for active RFIDs operating at 433.92 MHz clearly exceed the specified maximum distance. To compare the results to 13.56 MHz RFIDs as, e.g., used for contactless smartcards or Near Field Communication (NFC) cellphones, we conducted a similar study for according devices as well. The security problems resulting from our findings, including the unnoticed detection, duplication, and manipulation of transponders from a distance, demonstrate the need for suitable cryptographic protection to prevent such implementation attacks targeting the physical layer of the wireless interface. *The research in this section was published in [KOP11a].*

Mifare DESFire MF3ICD40

In order to illustrate that a smartcard implementing analytically secure algorithms like 3DES can still be susceptible to SCA, we focus on the Mifare DESFire MF3ICD40 in Chap. 7. This RFID, the secure variant of the broken Mifare Classic card, is used in several real-world installations, including large-scale public transport systems. Employing various state-of-the art methods and special filter circuits, we present full key-extraction attacks on this device, circumventing countermeasures against SCA. Applying the developed techniques for a case study of a real public transport system, we gain insight into the design of a practical application

and identify potential security issues in this regard. *The research in this section was published in [OP11, KOP11b].*

SimonsVoss Access Control System

According to our observations, proprietary cryptographic algorithms and protocols are still prevalent in modern electronic locking systems, which are used to control the access to both private and in particular corporate buildings. In Chap. 8, the SimonsVoss system 3060 "G2" is examined, applying reverse-engineering methods to recover the used undisclosed algorithms. Note that the vendor ignored SCA during the development and solely relied on obscurity-based countermeasures. Hence, a side-channel attack on the electronics of a lock can be used to extract a master key, compromising the security of an entire installation. *The research in this section was published in [OSS+13, SDK+13].*

The Yubikey One-Time Password Token

Two-factor authentication is becoming increasingly popular to solve the problems of traditional username-password credentials. We show that network operators using hardware tokens for this purpose have to take implementation attacks like SCA into account, a threat formerly irrelevant in the context of user authentication. To this end, we present a non-invasive side-channel attack on the Yubikey 2, a widely used USB token for generating one-time passwords. *The research in this section will be published in [ORP13].*

Maxim SHA-1 Product Authentication ICs

Hardware solutions are employed not only for authenticating users: ICs ensuring the genuineness of commercial devices, ranging from mass products to high-end equipment, are used to counter product piracy. Obviously, such components need to apply sound cryptographic algorithms to rule out "trivial" and analytical attacks. Yet, the threat due to SCA is given less attention. Analyzing two SHA-1-based ICs by Maxim Integrated, we found FI and SCA weaknesses, allowing to extract the full secret and subsequently "clone" the devices.

Altera FPGA Bitstream Encryption

In the context of product counterfeit, FPGA-based products face a special problem. Since the configuration bitstream of the FPGA usually resides in external memory, an adversary has direct access to a vendor's IP. To solve this problem, FPGAs comprise functionality to encrypt the configuration data. Nevertheless, these functions have been shown to be unprotected against implementation attacks for several FPGA families made by Xilinx and Actel. In Chap. 11, we demonstrate that the same problem exists for the third major FPGA vendor Altera. Having reverse-engineered the proprietary (AES-based) encryption scheme from the PC software Quartus II, successful key extraction attacks on Altera Stratix II FPGAs can be realized, allowing an adversary to clone, decrypt, and manipulate the FPGA's bitstream. *The research in this section was published in [MOPS13].*

Chapter 2

Side-Channel Analysis

Following the initial discovery of Simple Power Analysis and Differential Power Analysis in 1999, a significant number of improved techniques for SCA has been proposed. In this chapter, we give an overview over the most important results, describing possible sources of side-channel leakage and the most common evaluation methods to recover cryptographic secrets by means of SCA. We then focus on pre-processing techniques that have been shown to significantly improve the success rate of side-channel attacks. To this end, we propose a method to automatically derive (in a certain sense) optimal parameters for linear filters applied to SCA measurements.

Contents of this Chapter

2.1 Introduction

As introduced in Sect. 1.2, SCA can be used to extract cryptographic secrets irregardless of the cryptanalytical security of an algorithm by obtaining additional information on, e.g., an intermediate state via a physical side-channel.

The success of a side-channel attack depends on two main parts. First, the channel of the physical leakage has to be identified and captured in an adequate manner. To this end, Sect. 2.2 describes various well-known side-channels.

Second, suitable evaluation methods to reconstruct the targeted secret from the acquired measurements are needed. In Sect. 2.3, we focus on the—in practice still most common—techniques Simple Power Analysis (SPA), Differential Power Analysis (DPA), Correlation Power Analysis (CPA), and Template Attacks (TAs).

Often, the evaluation methods benefit (i.e., less measurements are needed to recover the secret) from additional (digital) pre-processing of the measurements. In Sect. 2.4, we accordingly summarize the use of Digital Signal Processing (DSP) techniques in the context of SCA. Subsequently, in Sect. 2.5, we detail on utilizing numerical optimization algorithms in order

to determine (in a certain sense) optimal parameters for linear pre-processing. This chapter is concluded in Sect. 2.6 with a summary of the main results.

2.2 Physical Side Channels

The actual physical side-channel leaking information on, e.g., a cryptographic secret, can take various forms. In this section, we give an overview over the physical side channels most often utilized in the literature, i.e., the timing (Sect. 2.2.1), the power consumption (Sect. 2.2.2), and the EM emanation (Sect. 2.2.3) of a DUT during the execution of a cryptographic algorithm. Additionally, in Sect. 2.2.4, we briefly describe more "exotic" leakage sources like the photonic emission of a semiconductor circuit.

2.2.1 Execution Time

In straightforward implementations of cryptographic algorithms, conditional statements can cause a dependency of the overall execution time on the processed data, possibly including the secret. For example, the basic Square-and-Multiply (SAM) algorithm given in Alg. 1 carries out an additional multiplication if a specific bit in the exponent is set. If Alg. 1 is used to sign a message with RSA, the runtime of the algorithm would thus partially be determined by the secret exponent d.

Algorithm 1 The left-to-right SAM algorithm for an RSA signature

Require: Basis x
Require: Modulus n
Require: w-bit exponent d
Ensure: $y = x^d \mod n$
  $y \leftarrow 1$
  for $i = w - 1$ to 0 **do**
    // Always square
    $y \leftarrow y^2 \mod n$
    // Exponent-dependent multiplication
    if bit i of d is set **then**
      $y \leftarrow y \cdot x \mod n$
    end if
  end for
  return y

In [Koc96], Kocher proposed an attack on the SAM algorithm with the additional requirement that the execution time of a single multiplication, e.g., $y \leftarrow y \cdot x \mod n$, varies depending on the operands x and y. Then, by predicting the runtime for the cases "bit $w-1$ of d set" and "bit $w-1$ of d not set" for varying inputs and comparing the prediction to measured runtimes (using the variance), bit $w-1$ of d can be recovered. This process is then repeated for each subsequent bit—taking the previously recovered bits into account—until the exponent has been completely determined.

A multitude of different timing attacks has been described in the literature following the initial discovery. A relatively recent example are cache-timing attacks [Ber]. These attacks are based on the fact that modern CPUs contain (multiple) fast caches. Accessing a specific value, e.g., from a Look-Up Table (LUT), for the first time is thus slower than subsequent accesses. The first time, the value is loaded from the slower Random Access Memory (RAM) (and stored in the cache), after that, it resides in the cache and can hence be retrieved faster.

When an S-box, e.g., of the AES, is implemented using a LUT, this introduces a key-dependent timing because the input to the S-box is the XOR of a plaintext and key byte (for the example of the AES). Thus, predicting the timing for all 256 candidates for a key byte and statistically comparing the prediction to the actual measurements allows to recover the full AES key. This attack was found to be applicable to the quasi-standard library `OpenSSL`—even over a network connection—in [Ber].

In this thesis, we did not directly focus on timing attacks, however, we observed two real-world examples of cryptographic algorithms with non-constant execution time for the DESFire MF3ICD40 (Chap. 7) and the Yubikey 2 (Chap. 9). Nevertheless, in both cases, we were unable to perform a timing attack. For other DUTs, e.g., Altera Stratix II FPGAs (Chap. 11) or Maxim SHA-1 ICs (Chap. 10), a hardware implementation of the employed algorithms lead to a constant, data-independent timing behavior.

2.2.2 Power Consumption

For an electronic circuit, the consumed power is not constant but depends on the processed data. For example, switching the input and thus the output of a Complementary Metal Oxide Semiconductor (CMOS) inverter leads to an increased current being drawn until the switching is "completed". Dynamic Random Access Memory (DRAM) memory stores information as the state of a capacitor (charged, not charged) and requires current for storing (and keeping) the "charged" state.

Thus, using the current consumption as a side channel to extract a secret from a cryptographic circuit seems to be a somewhat natural approach, and the first publications on SCA [KJJ99] were based on this leakage source. In the simplest case, the state of a key bit (set or not set) can be directly "read off" a current consumption measurement, otherwise, more complicated techniques are necessary, cf. Sect. 2.3.

A typical measurement setup for measuring the current consumption is depicted in Fig. 2.1. A (preferably small) resistor R is inserted into the ground line of the DUT and the voltage drop over the resistor is measured using a DSO or a similar digitizer. The resulting time and value-discrete current consumption waveform is usually referred to as *power trace* or simply *trace* in the literature.

Note that in the SCA research area, it is common to use the term "power", e.g., as in "power analysis" or "power trace", although technically the current consumption is measured in Fig. 2.1: The measured voltage $U_R = R \cdot I_{\mathrm{CC}}$ is proportional to the consumed current I_{CC}, while the power consumption of the DUT is

$$P_{\mathrm{DUT}} = I_{\mathrm{CC}} \cdot (U_{\mathrm{CC}} - U_R) = U_{\mathrm{CC}} \cdot I_{\mathrm{CC}} - R \cdot I_{\mathrm{CC}}^2$$

which contains an additional term proportional to the squared current I_{CC}^2 due to the resistor affecting the circuit. For an ideal current measurement, i.e., $R = 0$, the term with I_{CC}^2

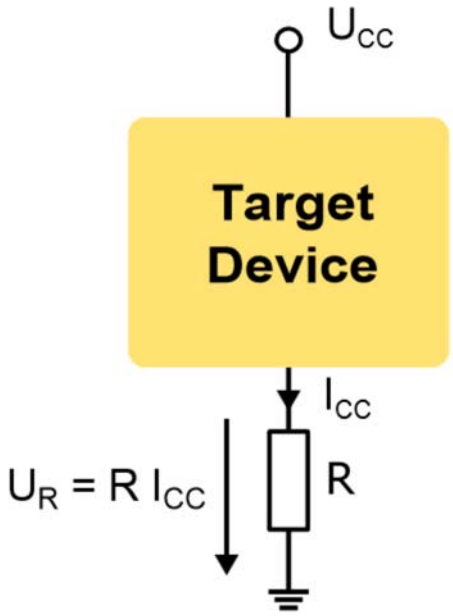


Figure 2.1: Typical setup for measurement of current consumption traces: a shunt resistor is inserted into the ground path

would vanish, and the measured current I_{CC} would indeed be exactly proportional to the power consumption $P_{\mathrm{DUT}} = I_{\mathrm{CC}} \cdot U_{\mathrm{CC}}$ for a constant supply voltage U_{CC}.

However, following the common practice, we occasionally use the term "power trace" or "measurement of the power consumption" also for the case $R > 0$ in this thesis, whereas a more correct expression would use "current". The term "power" can be understood as referring to a measurement involving the power supply lines of a circuit.

In general, in this thesis, we assume that the current consumption of a DUT at a point in time t has the form of Eqn. 2.1:

$$i\left(t\right) = I_{const} + i_{dyn}\left(t\right) \tag{2.1}$$

where I_{const} is the static part and $i_{dyn}\left(t\right)$ the fraction caused by internal operations, e.g., amongst others, intermediate values being manipulated during a cryptographic operation. Additionally, in reality, a noise term is added to Eqn. 2.1, caused both by measurement noise and algorithmic noise (e.g., interrupts, time desynchronization, and so on). This yields Eqn. 2.2:

$$i_{noisy}\left(t\right) = I_{const} + i_{dyn}\left(t\right) + \mathcal{N}\left(\mu_{noise},\, \sigma^2_{noise}\right) \tag{2.2}$$

where $\mathcal{N}\left(\mu_{noise}, \sigma^2_{noise}\right)$ is a normally distributed random variable with mean μ_{noise} and variance σ^2_{noise}.

The measurement of the current consumption has several disadvantages. First, it requires a modification of the circuit to insert the measurement resistor, except for devices with exposed ground and U_{CC} contacts like smartcards or simple USB tokens. For complex circuits, many ground lines may have to be cut, for example, to acquire traces for a µC. Second, blocking

and smoothing capacitors often have to be removed, a fact that makes an attack detectable afterwards.

For example, in the case of the SimonsVoss door lock Printed Circuit Board (PCB) (cf. Chap. 8), a large electrolyte capacitor led to heavily lowpass-filtered traces, rendering an attack based on the power consumption of the actually extremely SCA-vulnerable µC impossible. Finally, even worse, ICs might have on-chip voltage regulators that cause a similar smoothing and cannot be removed at all. This was for instance likely the case for the Yubikey 2 USB token, cf. Chap. 9.

2.2.3 Electro-Magnetic Emanation

In contrast to power measurements (Sect. 2.2.2), attacks based on the EM emanation [AARR03] usually require no modifications of the DUT or the surrounding circuitry of an IC. Figure 2.2 shows the typical measurement setup for EM analysis. A suitable EM probe is placed close to the DUT, whereas the exact position depends on the concrete attack scenario.

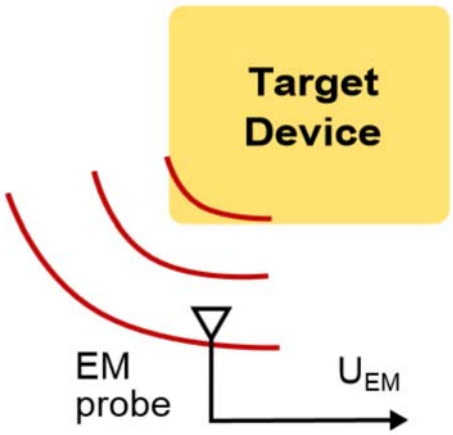


Figure 2.2: Typical setup for measurement of EM traces: an EM probe is placed close to the DUT

Depending of the type of the probe, either the electrical field E or the magnetic field H is primarily acquired. In addition, the exact nature of the leakage depends on the location of the probe. We describe the general cases of the probe close to the power supply pins and directly on the chip surface in Sect. 2.2.3 and Sect. 2.2.3, respectively. Besides, we focus on the special case of RFIDs—as a prerequisite of the analysis of the Mifare DESFire MF3ICD40 in Chap. 7—and outline methods to recover the side-channel signal in the presence of an RFID reader field.

Power Supply Pins

In many EM-based SCA attacks, relatively large magnetic probes are employed. For example, the probes used in this thesis (cf. Sect. 3.2.2) have a spatial resolution of a few millimeters, which is in the same range as the length of one side of a usual silicon die. In a Bachelor's thesis co-supervised by the author of the present thesis [Sch10], EM measurements with similar probes were compared to power traces. As a result, it turned out that such EM traces mainly

correspond to a wireless measurement of the DUT's current consumption. Accordingly, the best results were obtained for the EM probe placed on top of the supply voltage or ground pins of the DUT.

However, EM measurements have, apart from not requiring modifications of the DUT, the advantage that they can isolate the leakage of one specific IC on a larger PCB. As mentioned in Sect. 2.2.2, the power traces acquired for the SimonsVoss door lock analyzed in Chap. 8 were flattened by a capacitor and contained additional distortion from an ASIC not performing cryptographic operations. To isolate the side-channel leakage of the relevant µC, an EM probe on top of one of the ground pins of the µC yielded a signal suitable for SCA.

Chip Surface

In [Sch10], the wireless measurement of the power consumption was compared to traces acquired with an EM probe directly on the surface of the chip. To this end, the analyzed DUTs were decapsulated with fuming nitric acid to expose the silicon die. As a result, no location-dependent leakage could be identified, and side-channel attacks usually required a higher number of trace to succeed compared to direct and EM measurements of the supply lines.

A reason for this result may have been the insufficient spatial resolution of the employed probes. In [HMH+13], Heyszl et al. show that with high-resolution probes (approximately 100 µm resolution), the EM leakage of specific functional parts of an FPGA can be localized. These EM measurements lead to a significantly higher Signal to Noise Ratio (SNR) and thus reduce the number of required traces for SCA. As a side note, the authors of [HMH+13] mention that the leakage results from the magnetic component of the EM field, a statement that coincides with the findings in [Sch10].

RFID and Contactless Smartcards

For RFIDs, the energy for operation is supplied wirelessly using magnetic coupling. As proposed in [KOP09, KOP12], this gives rise to a different leakage mechanism compared to contact-based devices. In a similar manner as for regular data transmission, the 13.56 MHz field generated by the reader is load-modulated by the power consumption of an RFID, also cf. Chap. 6.3.1 for further details. However, for data transmission, the fluctuations of the EM field are intentional and far stronger in magnitude.

Thus, in an RFID setting, the amplitude of the reader signal is modulated by $i(t)$ (given in Eqn. 2.1), resulting in Eqn. 2.3 (neglecting the noise term of Eqn. 2.2). Note that in the case of RFID, the dynamic portion of the power consumption is far weaker than the static part, i.e., $|i_{dyn}(t)| << I_{const}$. The leakage exploitable for an SCA thus heavily depends on the quality of the isolation and amplification of $i_{dyn}(t)$.

$$s(t) = i(t) \cdot \cos(\omega_r \cdot t) = (I_{const} + i_{dyn}(t)) \cdot \cos(\omega_r \cdot t) \tag{2.3}$$

where $\omega_r = 2\pi\ f_r$, $f_r = 13.56$ MHz is the standard carrier frequency. Clearly, the extraction of $i(t)$ (and especially of the weak dynamic portion) from $s(t)$ can be done using amplitude demodulation, cf. for instance [SBS66]. In practice, "incoherent" techniques (i.e., for which a separate, unmodulated carrier signal is not necessary) based on rectification (often called envelope detection) are very common, and in this section, we follow that approach as well. The

principle due to which rectification can be used for demodulation is best understood in the frequency domain, following [Och06]. First note that, as stated above, $|i_{dyn}(t)| << I_{const}$ and hence

$$|s(t)| = |I_{const} + i_{dyn}(t)| \cdot |\cos(\omega_r \cdot t)| = (I_{const} + i_{dyn}(t)) \cdot |\cos(\omega_r \cdot t)|$$

Let $I(j\omega) = \mathrm{DFT}\{i(t)\} = \mathrm{DFT}\{I_{const} + i_{dyn}(t)\}$ denote the frequency domain representation of the signal that is to be reconstructed. By expanding $|\cos(\omega_r \cdot t)|$ using its Fourier series, one obtains the spectrum of the rectified signal:

$$\begin{aligned}\mathrm{DFT}\{|s(t)|\} &= \mathrm{DFT}\{i(t) \cdot |\cos(\omega_r \cdot t)|\} = \mathrm{DFT}\left\{i(t) \cdot \tfrac{2}{\pi} \textstyle\sum_{\nu=-\infty}^{\infty} \frac{(-1)^\nu}{1-4\nu^2} e^{j2\nu\omega_r t}\right\} \\ &= \tfrac{2}{\pi} \textstyle\sum_{\nu=-\infty}^{\infty} \frac{(-1)^\nu}{1-4\nu^2} \mathrm{DFT}\{i(t) \cdot e^{j2\nu\omega_r t}\} = \tfrac{2}{\pi} \textstyle\sum_{\nu=-\infty}^{\infty} \frac{(-1)^\nu}{1-4\nu^2} I(j\omega - j2\nu\omega_r)\end{aligned}$$

The rectified signal is essentially formed by the spectrum of $I_{const} + i_{dyn}(t)$, which, however, is (scaled and) repeated at all even multiples of the carrier frequency $\omega_r = 2\pi\ 13.56\,\mathrm{MHz}$. Thus, the first repetition occurs at 27.12 MHz as depicted in Fig. 2.3. Using a lowpass filter with a cutoff frequency less than 13.56 MHz isolates the desired signal $i(t)$. The constant term I_{const} can be removed with a highpass filter that only blocks the DC and very low-frequency components.

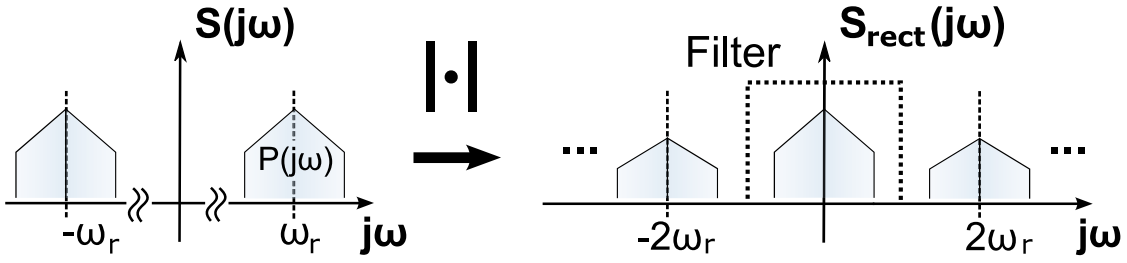


Figure 2.3: Full-wave rectification in the frequency domain

Note that in practice, often, the half-wave approach is used, discarding the negative part of $s(t)$. This approach has the advantage that it can be realized with one diode, whereas full-wave rectification requires more complex circuitry. The remaining half-period of the sine-shaped signal can be mathematically expressed as $\frac{1}{2}(s(t) + |s(t)|)$. In terms of the achievable bandwidth for receiving the modulating signal $i(t)$, full-wave rectification allows a maximum bandwidth of $B_{\mathrm{full}} = 13.56\,\mathrm{MHz}$, whereas the half-wave method limits it to $B_{\mathrm{half}} = {}^{13.56\,\mathrm{MHz}}/_{2}$ This is due to the fact that the resulting spectrum is essentially the sum of the spectrum of the original and the rectified signal, i.e.

$$\mathrm{DFT}\left\{\frac{1}{2}(s(t) + |s(t)|)\right\} = \frac{1}{2}(\mathrm{DFT}\{s(t)\} + \mathrm{DFT}\{|s(t)|\})$$

This results in a frequency-domain representation in which the first repetition occurs at $\omega_r = 2\pi\ 13.56\,\mathrm{MHz}$ as depicted in Fig. 2.4. In comparison to Fig. 2.3, the spectral components due to the addition of the non-rectified signal $s(t)$ (red, dashed in Fig. 2.4) require the lowpass

filter to have a cutoff frequency less than $^{13.56\,\mathrm{MHz}}/_{2}$ and hence lead to the mentioned reduction in bandwidth.

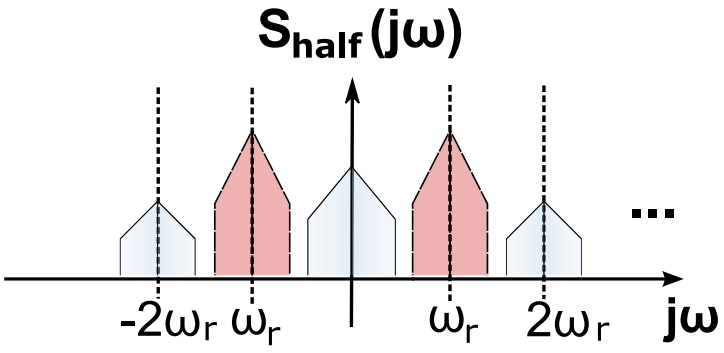


Figure 2.4: Half-wave rectification in the frequency domain. Red, dashed spectral components are due to the addition of the original signal

2.2.4 Other Side Channels

Apart from the classical side channels power consumption, EM emanation, and timing, a variety of more "exotic" leakage sources has been reported in the literature. In [ST04], it was shown that the acoustic emissions of a standard PC, e.g., caused by high-frequency vibrations of capacitors on the mainboard, can be used to perform timing attacks and distinguish CPU operations. In a similar direction, in [BDG+10] acoustic recordings were employed to reconstruct the text printed with a dot-matrix printer.

For semiconductors, Photonic Emission Analysis (PEA) enables the observation of active ICs down to the gate level [FH08, SNK+12]. By repeatedly taking photographs through a near-infrared microscope and averaging over hundreds of thousand runs, a complete map of the DUT's state at one point in time can be computed, with a resolution of approximately 1 µm in [SNK+12]. Besides, single-pixel detector can target a specific part of the DUT, e.g., an S-box output, to record the optical equivalent of a classical EM or power trace.

2.3 Evaluation Methods

To recover the key from measurements acquired via one of the physical side channels described in Sect. 2.2, again, a multitude of evaluation techniques has been proposed in the literature. Generally, these methods involve some form of DSP and statistical computations. In this section, we sum up the most important methods which are employed throughout this thesis, i.e., SPA, DPA, CPA, and TA. Note that we use the term "power analysis" independent of the actual side channel and do not distinguish, e.g., between a DPA and a Differential Electro-Magnetic Analysis (DEMA), as it is sometimes done in the literature.

2.3.1 Simple Power Analysis

In an SPA [KJJ99], one or an average over several traces (with identical input) to reduce the noise is—most often visually—inspected to directly derive a cryptographic secret. For example, considering Alg. 1, a multiplication could be distinguishable from a squaring in a trace. Thus, when using the SAM algorithm to compute an RSA signature, the secret exponent could directly be read off the trace.

SPA can apply to a variety of cryptographic algorithms, including the AES key schedule [Man02] and ECC scalar multiplication [Wal04]. In this thesis, SPA is mostly used when profiling a DUT with an unknown implementation to prepare a key recovery. For instance, cryptographic operations often lead to an increased power consumption and can thus be identified in a trace, reducing the amount of points in time to be evaluated.

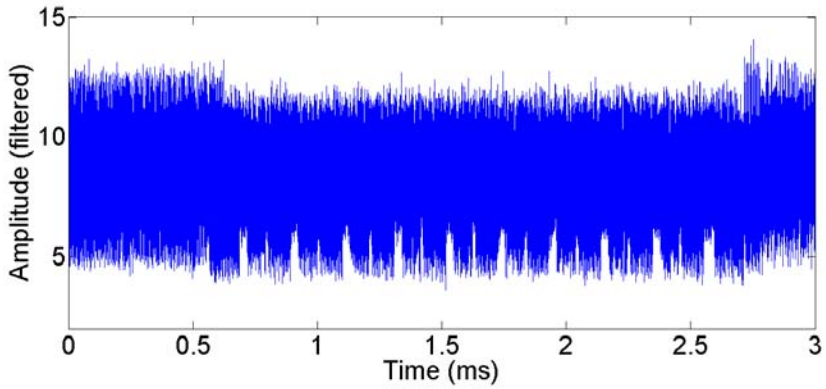


Figure 2.5: Example of an EM trace for the Yubikey 2, showing the ten AES rounds

An example for an application of SPA for profiling purposes is given in Fig. 2.5. The depicted EM trace for the AES on the Yubikey 2, cf. Chap. 9, allows to draw several conclusions on the details of the implementation. First, it can be seen that the AES takes approximately 2.2 ms to execute, whereas each of the ten rounds is clearly distinguishable. In accordance with the AES description, the tenth round is shorter, as it misses the `MixColumns` step. Zooming further into the trace, additional information, e.g., the approximate clock frequency of the DUT could be recovered. To this end, further details for the profiling of the Yubikey 2 can be found in Sect. 9.4.

2.3.2 Differential Techniques using Statistics: Differential Power Analysis and Correlation Power Analysis

In contrast to SPA, DPA [KJJ99] and CPA [BCO04] are based on the evaluation of many traces with varying input data for the targeted algorithm. Then, a brute-force attack with additional information is performed on a part of the algorithm. For instance, a DPA attack on the DES or AES usually only targets the first round, for which each S-box is processed separately. Hence, each subkey entering one specific S-box can be recovered independently from the remaining key bits. By testing all possible candidates for a given subkey (e.g., $2^6 = 64$ for the DES or $2^8 = 256$ for the AES), the full key can be extracted in a divide-and-conquer manner.

The key idea behind DPA and related methods is that side-channel traces can be used to identify the correct subkey amongst the candidates. A statistical test is employed to determine which subkey candidate is most likely to have caused the observed leakage. This is where according evaluation methods differ. For example, DPA is based on computing the difference between two trace sets, while CPA uses the correlation coefficient [BCO04] to perform this test for dependency.

The process of performing a DPA or CPA can be divided into two steps. In the measurement phase, the adversary has physical access to the DUT and records some side-channel signal (e.g., the power consumption or the EM emanation during the cryptographic computation) that is related to the processed data. This step is repeated N times with varying input data M_i, yielding N time-discrete waveforms $x_i(t)$ with T points each.

In the evaluation phase, the key is recovered by fixing a (small) subset $\mathcal{K}_{cand} \subseteq \mathcal{K}$ and considering all key candidates $k \in \mathcal{K}_{cand}$: for each $k \in \mathcal{K}_{cand}$ and for each $i \in \{0, \ldots, N-1\}$, a hypothesis $V_{k,i}$ on the value of some intermediate is computed. This hypothesis could, for example, be the output of one AES S-box in the first round given in Eqn. 2.4:

$$V_{k,i} = \mathrm{SBox}\left(M_i \oplus \left(\hat{K} = k\right)\right) \tag{2.4}$$

Differential Power Analysis

For a DPA, first, one bit of the intermediate $V_{k,i}$ is selected. Then, for each key candidate $k \in \mathcal{K}_{cand}$, $V_{k,i}$ is computed for each trace and the traces are assigned to one of two sets $D_0^{(k)}$ and $D_1^{(k)}$. If the selected bit in $V_{k,i}$ is set, the respective trace $x_i(t)$ is included in $D_1^{(k)}$, otherwise in $D_0^{(k)}$. Then, the average traces for both sets are separately computed as

$$\bar{x}_0^{(k)}(t) = \frac{1}{\left|D_0^{(k)}\right|} \sum_{x(t) \in D_0^{(k)}} x(t) \text{ and } \bar{x}_1^{(k)}(t) = \frac{1}{\left|D_1^{(k)}\right|} \sum_{x(t) \in D_1^{(k)}} x(t). \tag{2.5}$$

Finally, the Difference of Means (DoM)

$$\delta_{DPA}^{(k)}(t) = \bar{x}_0^{(k)}(t) - \bar{x}_1^{(k)}(t) \tag{2.6}$$

is computed. The correct key candidate $\hat{k}$ is then the one for which $\delta_{DPA}^{(\hat{k})}(t)$ exhibits the maximum DoM value. The intuition behind DPA is the following: if the grouping according to the selected bit is correct, i.e., the correct value $\hat{k}$ is used, traces with lower (bit cleared) and higher (bit set) values are averaged separately. The difference of the averages then shows peaks for the time instants at which the DUT processes the respective intermediate. For a wrong grouping, the traces (with the bit set or cleared) are randomly assigned to either set, so the difference averages out and the DoM is "flat".

Correlation Power Analysis

In a CPA, several bits can be taken into account to better model the physical process behind the leakage and thus to reduce the number of required traces compared to a DPA. Using a power model f, the hypothesis $V_{k,i}$ on the value of some intermediate value is mapped to $h_{k,i} = f(V_{k,i})$ to describe the process that causes the side-channel leakage. In practice, for

DUTs such as FPGAs or µCs, the power model is often either the Hamming Weight (HW) or Hamming Distance (HD) model [MOP07].

$h_{k,i}$ and $x_i(t)$ are treated as observations of discrete random variables. In order to detect the dependency between $h_{k,i}$ and $x_i(t)$, the correlation coefficient $\rho_k(t)$ (for each point in time $t \in \{0, \ldots, T-1\}$ and each key candidate $k \in \mathcal{K}_{cand}$) is given by Eqn. 2.7:

$$\rho_k(t) = \frac{cov(x(t), h_k)}{\sqrt{var(x(t))\, var(h_k)}} \tag{2.7}$$

with $var(\cdot)$ indicating the sample variance and $cov(\cdot, \cdot)$ the sample covariance according to the standard definitions [Wei10]:

$$\begin{aligned}
cov(x(t), h_k) &= \frac{1}{N-1} \sum_{i=0}^{N-1} (x_i(t) - \bar{x}(t)) \left(h_{k,i} - \bar{h}_k\right) \\
\bar{x}(t) &= \frac{1}{N} \sum_{i=0}^{N-1} x_i(t) \\
\bar{h}_k &= \frac{1}{N} \sum_{i=0}^{N-1} h_{k,i} \\
var(x(t)) &= \frac{1}{N-1} \sum_{i=0}^{N-1} (x_i(t) - \bar{x}(t))^2 \\
var(h_k) &= \frac{1}{N-1} \sum_{i=0}^{N-1} \left(h_{k,i} - \bar{h}_k\right)^2
\end{aligned}$$

As for a DPA, the key candidate $\hat{k}$ with the maximum correlation $\hat{k} = \arg\max_{k,t} \rho_k(t)$ is assumed to be the secret key k^{dut} used by the DUT. Note that CPA covers a single-bit DPA as a special case. When $f(V_{k,i})$ is chosen to select a specific bit of the intermediate, Eqn. 2.7 reduces to a form equivalent to the DoM of Eqn. 2.6 apart from a constant normalization factor [MOP07].

The use of the correlation coefficient in a CPA has several practical advantages. First, the value of the correlation is bounded by ±1, i.e., $-1 \leq \rho_k(t) \leq 1$. This allows to compare the amount of leakage for different DUTs or the quality of different measurement setups. Moreover, several useful shorthand formulas exist that are helpful when assessing CPA results [MOP07]. For instance, given a specific correlation coefficient value $\hat{\rho} \in [-0.2, 0.2]$, the number of required traces to identify the correct key candidate in this case can be estimated using Eqn. 2.8.

$$\#\,\text{required traces} = \frac{28}{\hat{\rho}^2} \tag{2.8}$$

Besides, an expected "noise level" for a given number of traces can be derived. For wrong key candidates, the correlation coefficients follow a Gaussian distribution with standard deviation $\sigma = 1/\sqrt{\#\,\text{traces}}$. 99.99% of the samples taken from this distribution are within $\pm 4\sigma$, which yields the boundary for the expected maximum correlation ρ_{wrong} for wrong candidates of Eqn. 2.9:

$$|\rho_{\text{wrong}}| \leq \frac{4}{\sqrt{\#\,\text{traces}}} \tag{2.9}$$

Throughout this thesis, this noise level is included in plots of the correlation coefficient and indicated by blue horizontal lines.

2.3.3 Template Attacks

In contrast to CPA, a TA requires a profiling phase, i.e., a step during which the DUT is under full control of the adversary to estimate the statistical relation between the observable random variables—in our case the respective points in time of a trace—and the internal states that are to be distinguished (for example, the value of a key byte). The resulting *training set* $\mathcal{S}^{training}$ is then used to recover the desired values from a *test set* $\mathcal{S}^{test}$, i.e., traces for which the value of the key byte is considered unknown.

Algorithm 2 Template creation and matching procedure for a key byte

for $b = 0 \dots 255$ **do**
  {Estimate sample mean and covariance matrix from training set for byte b}
  $(\vec{\mu}_b,\, \Sigma_b) \leftarrow \text{estimate}\left(\mathcal{S}_b^{training}\right)$
end for
{Estimate average covariance matrix for the training data}
$\overline{\Sigma} \leftarrow \frac{1}{256} \sum_{b=0}^{255} \Sigma_b$
{Estimate sample mean and covariance matrix for the test data}
$(\vec{\mu}',\, \Sigma') \leftarrow \text{estimate}\left(\mathcal{S}^{test}\right)$
{Compute distance between the test and training data}
for $b = 0 \dots 255$ **do**
  $\delta_b \leftarrow \text{distance}\left(\vec{\mu}_b,\, \Sigma_b,\, \overline{\Sigma},\, \vec{\mu}',\, \Sigma'\right)$
end for
return $\underset{b}{\text{argmin}}\, \delta_b$

Assume a TA is used for recovering the value of an unknown key byte, i.e., to identify the correct value amongst the 256 candidates. In the profiling phase, for each candidate value $0 \leq b \leq 255$, N_P traces $\vec{x}_{b,\,i}$ of length T form the training set $\mathcal{S}_b^{training} = \{\vec{x}_{b,\,1},\, \dots,\, \vec{x}_{b,\,N_P}\}$ for a specific candidate b.

Then, to extract the correct value from measurements where the key is unknown, the traces are compared to all candidates b using the respective $\mathcal{S}_b^{training}$. For a thorough evaluation of a TA, one usually tests how well all 256 possible values can be recovered, again yielding—for each b—a test set $\mathcal{S}_b^{test} = \left\{\vec{x}'_{b,\,1},\, \dots,\, \vec{x}'_{b,\,N_P}\right\}$ Given $\mathcal{S}^{test}$ for a fixed but unknown key—in our case, the test set for some key byte value b—the comparison to the training data is carried out as outlined in Alg. 2.

$\text{estimate}\,(\cdot)$ is an algorithm that estimates the (pointwise) sample mean and covariance matrix from the respective set of traces, e.g., using the standard empirical formulas, cf. Sect. 2.3.2. $\text{distance}\,(\cdot)$ is a suitable distance measure based on the previously estimated statistical parameters. The value for the key byte b that minimizes the chosen distance measure is then returned as the most probable candidate for the given test traces. We exemplarily selected the following distance measures:

Difference of Means The simplest case only evaluates the norm of the pointwise difference of the class means, discarding any information on the (co-)variances:

$$\sum_{t=1}^{T} \left(\vec{\mu}_b\left(t\right) - \vec{\mu}'\left(t\right)\right)^2$$

Euclidean Assuming that the covariance matrix is diagonal, one obtains the Euclidean distance for which the differences are normalized using the pointwise variance

$$\sum_{t=1}^{T} \frac{\left(\vec{\mu}_b\left(t\right) - \vec{\mu}'\left(t\right)\right)^2}{\Sigma_b\left(t,\, t\right)}$$

Mahalanobis Taking all available parameters of the distribution into account, the Mahalanobis distance [Mah36] is given as

$$\left(\vec{\mu}_b - \vec{\mu}'\right)^T \overline{\Sigma^{-1}} \left(\vec{\mu}_b - \vec{\mu}'\right)$$

Using this measure with an identical covariance matrix $\overline{\Sigma}$ for all classes is equivalent to the "standard" template probability formula [CRR02, MOP07], since the constant factor and the continuous exponential function does not affect the assignment of a sample to a class (high probability leads to low distance):

$$p_b\left(\vec{\mu}'\right) = \frac{1}{\sqrt{\left(2\pi\right)^T det\, \overline{\Sigma}}} \exp\left(-\frac{1}{2}\, \left(\vec{\mu}_b - \vec{\mu}'\right)^T \overline{\Sigma^{-1}} \left(\vec{\mu}_b - \vec{\mu}'\right)\right)$$

Note that in this equation, the $\left(2\pi\right)^T$ in the denominator indicates an exponentiation with the length of a trace T, while the $\left(\cdot\right)^T$ in vector and matrix operations refers to a transpose.

A practical application of TAs can be found in Chap. 7, where templates for the transfer of a secret over an internal data bus are used to extract the key of the Mifare DESFire MF3ICD40. However, the analysis of the DESFire MF3ICD40 also demonstrates a significant problem of TAs. In contrast to DPA or CPA, where the *relative* differences of traces from one measurement run are evaluated, a TA compares the *absolute* values of two separate measurement runs for two instances of a DUT. Hence, even small variations, e.g., due to imperfections of the manufacturing process, can render a TA impossible.

Thus, the question about the "portability" of templates arises, i.e., how well the measurements for the training device match the attacked DUT. In [EG12], it was shown that a significant amount of additional processing is needed to apply templates recorded with a DES ASIC to the identical ASIC after changes in the measurement setup and aging of the DUT. In [RSVC+11], the "portability" is formalized using the notion of perceived information. Still, a general solution for the problem of applying the profiling templates to a different test device has not been proposed to our knowledge.

2.3.4 Other Methods

Apart from the standard methods SPA, DPA/CPA, and TA, numerous other evaluation methods are available. These can be subdivided into several classes:

One class aims at replacing the statistical test, e.g., the DoM or the correlation coefficient, with other methods to relax the assumptions for the model of the DUT's leakage characteristics. In this regard, Mutual Information Analysis (MIA) proposed by Gierlichs et al. [GBTP08] is the most established technique. In contrast to CPA, MIA can capture non-linear dependencies between the predicted power consumption and the measured values and hence improve the success rate of SCA in certain situations.

A different kind of evaluation method, termed "horizontal" power analysis in [CFG+10], is based on detecting and utilizing correlations within a single trace, e.g., to identify the processing of similar values in a cryptographic algorithm. These methods apply both to symmetric and asymmetric cryptographic primitives. For example, in [CFG+10], according techniques are employed to extract the secret exponent of an RSA exponentiation from a single trace. For symmetric cryptography, similar methods usually called collision attacks have been proposed [SLFP04, Bog07]. In this case, an adversary detects identical input values for different S-box lookups and can then set up equations to derive the secret key.

Other methods include Combined Implementation Attacks (CIA), for which a side-channel attack is combined with FI, e.g., to disable certain countermeasures or cause specific leakage patterns [AVFM07, STA+10], and Algebraic Side-Channel Analysis (ASCA) [RS09], for which the description of the target algorithm and the observed leakages are combined in a system of equations. These equations are then solved with suitable algorithms to recover secret information.

2.4 Pre-Processing Methods

Pre-processing techniques are widely used to increase the success rate of side-channel analysis when attacking (protected) implementations of cryptographic algorithms. One of the first examples of DSP being applied to SCA can be found in [MDS99]: the authors mention the use of a matched filter to increase the SNR and thus the height of a DPA peak. In [CCD00], Clavier et al. proposed to perform comb filtering, i.e., average measurement samples from multiple clock cycles in order to increase the success rate of a DPA in the presence of random process interrupts.

Especially for practical attacks on cryptographic devices, DSP pre-processing is often mandatory, for instance because of uncorrelated noise due to non-cryptographic parts of an IC. In [BPT10], digital filtering helped to isolate the side-channel leakage of a cryptographic coprocessor. Similarly, the attacks on the bitstream encryption mechanism of Xilinx FPGAs required the removal of an interfering signal [MBKP11].

For SCA utilizing the EM emanation of a cryptographic device, digital filters have been applied to isolate the frequencies containing the side-channel leakage [AARR03]. In the context of cryptographic RFID devices, DSP pre-processing steps have been shown to be necessary [PHF08]. Accordingly, the attacks on the Mifare DESFire MF3ICD40 RFID smartcard involve several filter operations, cf. Chap. 7.

2.4.1 Linear Filters

A CPA in the time domain is thus often preceded by a linear Finite Impulse Response (FIR) filter: for example, a bandpass or bandstop filter may be used to isolate or remove certain frequencies present in a side-channel signal [BPT10]. Integrating over multiple clock cycles can be interpreted as a comb filter [CCD00].

An $S-1$-th order FIR filter is defined by S coefficients $a_i \in \mathbb{R}$, $i = 0 \ldots S-1$. The response $y(t)$ of the filter to the input signal $x(t)$ is computed as a (sliding) weighted sum of the points of the input signal as in Eqn. 2.10.

$$y(t) = \sum_{i=0}^{S-1} a_i x(t-i) \tag{2.10}$$

In Sect. 2.5, we show how to compute the correlation coefficient between a prediction h_k and an arbitrary weighted sum like an FIR filter, given the "raw" correlation for each point in time and the covariance matrix of the input signal. To this end, we apply a matrix notation according to [HSST03]. Throughout this thesis, digital filters are applied in various cases to improve SCA attacks, e.g., for the Mifare DESFire MF3ICD40 in Chap. 7, the SimonsVoss lock in Chap. 8, the Yubikey 2 in Chap. 9, Maxim SHA-1 ICs in Chap. 10, and Altera Stratix II FPGAs in Chap. 11.

2.4.2 SCA in the Frequency Domain

To overcome misalignment of traces and to isolate a leakage signal in a specific frequency band, a method termed Differential Frequency Analysis (DFA) was proposed in [GHT05] and shown to be beneficial in [PHF08]. DFA can be seen as a (non-linear) pre-processing step performed before a CPA or DPA: First, each trace is partitioned into (overlapping) segments or "windows" of length w_{in}. These windows are transformed to the frequency domain with the Discrete Fourier Transform (DFT), and the phase information is discarded by taking the absolute value of the DFT coefficients.

The central parameter that determines the effectiveness of DFA is the size of each window w_{in}. If the window is chosen too wide, the DFT "averages" over various operations or clock cycles of the DUT and the usable amount of leakage decreases. If the window is too narrow, not all information belonging to the targeted operation is taken into account. Finding a suitable value for w_{in} is usually done based on *(i)* information such as the clock frequency of the DUT and *(ii)* experiments with various values for w_{in}, selecting the window size that maximizes the correlation.

Apart from that, to increase the spectral resolution, the resulting windows can be padded with zeroes, a standard technique in DSP [Smi97]. To reduce the number of points for the subsequent DPA or CPA, all frequency points or "bins" except those for which the correlation occurs can be left out. For a zero-padded window of size $w_{win} = w_{in} + w_{pad}$ recorded at a sample rate of f_s, the frequency bin n_{leak} corresponding to a specific frequency f_{leak} is computed as:

$$n_{leak} = \left\lfloor \frac{f_{leak}}{f_s} \cdot w_{win} \right\rfloor$$

For this thesis, we employed DFA in cases where an alignment of the traces in the time domain could not be carried out using pattern matching techniques [MOP07]. For both the

Mifare DESFire MF3ICD40 (Chap. 7) and the Altera Stratix II FPGA (Chap. 11), the leakage signal could not be directly identified in the time domain. Hence, we used DFA to significantly improve the success rate of the respective attacks.

2.5 Finding Optimal Linear Transforms for SCA

In this section, we present an analytical expression for the correlation coefficient after applying a linear transform to the side-channel traces. Doing so, we are able to precisely quantify the influence of a linear filter on the result of a CPA. On this basis, we demonstrate the use of optimization algorithms to efficiently and methodically derive "optimal" filter coefficients in the sense that they maximize a given definition for the distinguishability of the correct key candidate. We verify the effectiveness of our methods by analyzing both simulated and real-world traces for a hardware implementation of the AES.

As mentioned in Sect. 2.4, methods such as DPA or CPA often benefit from prior signal processing, e.g., FIR filtering. Yet, the precise effect of the pre-processing steps on the success rate of SCA has, to our knowledge, not been precisely quantified. In this section, we utilize analytical properties of the correlation coefficient in order to derive a more systematic approach for optimizing the—so far mostly heuristically selected—pre-processing parameters. From a designer's point of view, our method helps to objectively estimate the amount of leakage an adversary might extract by means of filtering.

There is almost no systematic research how DSP operations such as filtering affect the outcome of SCA. In [BPT11], the authors propose an approach to automatically determine appropriate bandpass filters, using CPA as a block algorithm that is repeatedly executed for different choices for the filter coefficients. In general, however, filtering is seen as a completely separate pre-processing step, and the parameters are usually chosen manually.

The technique developed in this chapter was published at CARDIS 2012 [OP13]. This research was mainly motivated by the various real-world side-channel attacks described in Chap. 7–11, which often require digital pre-processing to succeed.

2.5.1 Contribution

The remainder of this section is organized as follows: We introduce a matrix notation that provides a closed form for the correlation coefficient after a linear transform in Sect. 2.5.2. On this basis, we propose the use of numerical optimization to determine "good" filter coefficients in Sect. 2.5.3. In Sect. 2.5.4 and Sect. 2.5.5, we compare our approach to normal CPA and to a CPA in the frequency domain, using simulated and real-world measurements (provided in the second DPA contest [COM]), respectively. Finally, we conclude in Sect. 2.6.

2.5.2 Matrix Notation

To improve the readability, we drop the index k for the key candidate in this section. All involved quantities are represented as vectors or matrices. A trace $x_i(t)$ is hence denoted as a $T \times 1$ vector $\vec{x_i}$. $\Sigma_{\vec{x}\vec{x}}$ is the $T \times T$ sample covariance matrix over all traces according to the standard definition.

For the purposes of this work, the prediction is a scalar h_i, i.e., a 1×1 vector. Note that this restriction is not mandatory: the prediction could also be extended to a $P \times 1$ vector $\vec{h}_i$, for example, to handle multiple bits of one prediction separately. Again, $\Sigma_{\vec{h}\vec{h}}$ is the $P \times P$ covariance matrix. For the scalar case $P = 1$, this reduces to the usual variance.

Finally, $\Sigma_{\vec{x}\vec{h}}$ is the $T \times P$ covariance matrix between $\vec{x}$ and $\vec{h}$. For $P = 1$, this corresponds to the covariance term in the denominator of the traditional formula for the correlation coefficient. Then, given a $T \times 1$ weight vector $\vec{a}$ and a $P \times 1$ weight vector $\vec{b}$, a *closed form* for the correlation coefficient between the dot products $\vec{a} \cdot \vec{x}_i$ and $\vec{b} \cdot \vec{h}_i$ is given by Eqn. 2.11 [HSST03].

$$\rho_{\vec{x}\vec{h}}\left(\vec{a},\, \vec{b}\right) = \frac{\vec{a}^T \cdot \Sigma_{\vec{x}\vec{h}} \cdot \vec{b}}{\sqrt{\vec{a}^T \cdot \Sigma_{\vec{x}\vec{x}} \cdot \vec{a}}\sqrt{\vec{b}^T \cdot \Sigma_{\vec{h}\vec{h}} \cdot \vec{b}}} \tag{2.11}$$

This representation is fully equivalent to performing the dot products first (as a pre-processing step) and then computing the correlation coefficient on the pre-processed data. In particular, $\vec{a}$ can be seen as the coefficients of an FIR filter that is applied to each trace separately. Similarly, $\vec{b}$ corresponds to an arbitrary weighted sum of, e.g., several bits of a predicted intermediate. As stated above, for the purposes of this work, we assume a scalar prediction h_i and hence set $P = 1$ and $b = 1$ for the remainder of this section. As a side note, in order to obtain the unfiltered correlation coefficient at time index $t = 0, 1, \ldots$, only the t'th entry of $\vec{a}$ has to be set to a non-zero value, i.e., $\vec{a} = (1\,0\,0\,\ldots\,0)\,, (0\,1\,0\,\ldots,0)\,,$ and so on.

Note that Eqn. 2.11 can be naturally extended to incorporate a *transform matrix* rather than a vector. In this case, $\vec{a}$ becomes a $T \times S$ matrix A formed by S different column vectors $\vec{a}_s$, $s \in 0, \ldots S-1$. The $S \times 1$ correlation coefficient vector $\vec{\rho}_{\vec{x}\vec{h}}\left(A,\, \vec{b}\right)$ is then given by evaluating Eqn. 2.11 for all $\vec{a}_s$ and concatenating the results. This form covers (amongst other linear transforms) any FIR filter: A consists of the filter coefficient vector that is shifted by s positions for row s, that is, A is a Toeplitz matrix [Smi07].

Computational Complexity

The form of Eqn. 2.11 is useful when the correlation coefficient is to be computed for fixed $\vec{x}_i$ and h_i but for (many) different $\vec{a}$: first computing $\vec{a} \cdot \vec{x}_i$ for each trace and then evaluating the traditional formula for the correlation coefficient has a complexity of $\mathcal{O}(N\,T)$. For L different choices $\vec{a}^l$, $l = 0 \ldots L-1$, the overall effort is thus $\mathcal{O}(L\,N\,T)$. In contrast, to evaluate Eqn. 2.11, the computation of the covariance matrices needs $\mathcal{O}\left(N\ (T^2 + T)\right) = \mathcal{O}\left(N\,T^2\right)$ operations. The post-processing to obtain the desired correlation coefficients for all $\vec{a}^l$ is then $\mathcal{O}\left(L\,T^2\right)$ (which is independent of N). Hence, the total complexity is $\mathcal{O}\left(N\,T^2 + L\,T^2\right)$.

Complexity Reduction in Special Cases

The main drawback of Eqn. 2.11 is that it requires the covariance matrix $\Sigma_{\vec{x}\vec{x}}$. For large T, the estimation of this matrix is problematic due to issues with the computational complexity. Thus, the application of the closed form of the correlation may become infeasible. Incidentally, the statistical efficiency is not an issue in this case—Equation 2.11 does not involve the inverse of $\Sigma_{\vec{x}\vec{x}}$ and is fully valid for small sample size N.

Still, for a filter of order $S - 1$, $\vec{a}$ only has S consecutive non-zero coefficients. Hence, in this case, it is sufficient to estimate $\Sigma_{\vec{x}\vec{x}}$ as a band matrix with a bandwidth of S. The

computational complexity is then reduced to $\mathcal{O}(ST)$, i.e., linear with respect to the length of the traces. Finally, for the optimization approach proposed in the following Sect. 2.5.3, $\Sigma_{\vec{x}\vec{x}}$ can be factored out when determining $\vec{a}$. Hence, if estimating $\Sigma_{\vec{x}\vec{x}}$ becomes prohibiting for large T, one may still utilize the traditional approach and compute the dot products $\vec{a} \cdot \vec{x}_i$ before the CPA once the optimal $\vec{a}$ has been found.

2.5.3 Optimal Linear Transforms for CPA

Given the closed-form expression for the transformed correlation coefficient of Eqn. 2.11, we propose a method to find "optimal" filter coefficients $\vec{a}$ in the sense that the filter maximizes the distinguishability of the correct key candidate.

To achieve this goal, we regard Eqn. 2.11 as a multivariate function in $\vec{a}$ and employ standard numerical optimization algorithms. As we aim to maximize the distinguishability rather than the correlation itself, a suitable optimization criterion has to be defined. We assume a semi-profiled scenario in which an adversary possesses an instance of the DUT for which he knows the secret key. Note that the adversary is not necessarily able to change the key—only the knowledge of the correct key is required for the optimization of the filter coefficients. In contrast to TAs, which have been shown to be sensitive to process variations, cf. Sect. 2.3.3, we expect the filter coefficients to be less sensitive in this regard. This conjecture is based on the fact that a filter modifies the frequency spectrum (which should be less device-dependent than, e.g., the signal amplitude), while the actual key recovery is still carried out by a (more robust) differential technique like CPA.

In our experiments, directly maximizing Eqn. 2.11 gave rise to overfitting of the coefficients $\vec{a}$. As a result, the correlation is maximized for one specific problem instance (i.e., fixed input data, key, and traces), however, if any parameter changes, the determined coefficients no longer lead to the desired result. Hence, we devised the criterion given in Eqn. 2.12. The goal is to maximize the ratio between the absolute value of the correlation coefficient for the correct key k^{dut} and the average over the absolute value of the correlation coefficients for incorrect key candidates $k_{wrong} \in \mathcal{K}_{optim}$, $\mathcal{K}_{optim} = \mathcal{K}_{cand} \setminus \left\{k^{\text{dut}}\right\}$.

$$f_{objective}(\vec{a}) = \frac{\left|\rho_{\vec{x}h_{k^{\text{dut}}}}(\vec{a})\right|}{1/|\mathcal{K}_{optim}| \left(\sum_{k \in \mathcal{K}_{optim}} \left|\rho_{\vec{x}h_k}(\vec{a})\right|\right)} \tag{2.12}$$

Note that in Eqn. 2.12, every $\rho_{\vec{x}h}$ both in the numerator and denominator contains a positive factor of $1/\sqrt{\vec{a}^T \cdot \Sigma_{\vec{x}\vec{x}} \cdot \vec{a}}$ (independent of h_k) which can be canceled. Equation 2.12 thus takes the form of Eqn. 2.13 (whereas the factor $1/|\mathcal{K}_{optim}|$ was left out).

$$f_{objective}(\vec{a}) = \frac{\left|1/\sqrt{\Sigma_{h_{k^{\text{dut}}} h_{k^{\text{dut}}}}} \cdot \vec{a}^T \cdot \Sigma_{\vec{x}h_{k^{\text{dut}}}}\right|}{\sum_{k \in \mathcal{K}_{optim}} \left|1/\sqrt{\Sigma_{h_k h_k}} \cdot \vec{a}^T \cdot \Sigma_{\vec{x}h_k}\right|} \tag{2.13}$$

This eliminates the computationally most expensive part of Eqn. 2.11, namely the vector-matrix product with complexity $\mathcal{O}(T^2)$. Moreover—at least in the profiling step—the covariance matrix $\Sigma_{\vec{x}\vec{x}}$ is not needed at all. Hence, as mentioned in Sect. 2.5.2, the optimization of the weight coefficients can be carried out even with long traces, for which computational issues make the estimation of the sample covariance matrix difficult or impossible. To numerically

find an optimum of $f_{objective}$, we employed the function `fminunc` provided by the MATLAB optimization toolbox [The]. This function *minimizes* a given objective function. In our case, we thus search for a minimum of $-f_{objective}$ (which is equivalent to a maximum of $f_{objective}$).

Relation to Other Techniques

Principal Component Analysis The method of Principal Component Analysis (PCA) [SA08] transforms signals to a new (lower-dimensional) representation. Recently, Batina et al. proposed to use PCA as a pre-processing step for a CPA [BHvW12]. Their idea is based on the observation that a leakage signal and unrelated noise are often mapped to different principal components. PCA is a linear transform, i.e., a trace $\vec{x}_i$ is projected to the new representation using the vector-matrix product $\vec{y} = U^T \cdot \vec{x}_i$ with U the matrix of the (retained) eigenvectors of the covariance matrix $\Sigma_{\vec{x}\vec{x}}$. Thus, as mentioned in Sect. 2.5.2, one point of the projected trace $\vec{y}$ is given as the scalar product between $\vec{x}_i$ and one row of U^T. The rows of U^T can therefore be regarded as different choices for the weight vector.

Canonical Correlation Analysis In contrast to PCA which picks principal components with maximum variance, Canonical Correlation Analysis (CCA) [HSST03] finds a weight vector that maximizes the correlation coefficient. Performing an eigenvector decomposition of the covariance matrix, CCA finds a vector $\vec{a}$ that maximizes Eqn. 2.11. However, as mentioned in Sect. 2.5.3, in our experiments this led to overfitting and resulted in non-applicable weight vectors.

SCA in the Frequency Domain Transforming traces to the frequency domain and discarding the phase component, i.e., using DFA, has been shown to be beneficial for SCA, cf. Sect. 2.4.2. This pre-processing step is both applicable to overcome misalignment of the traces and to spectrally isolate the leakage component. Note that the phase component is removed by taking the absolute value of the transformed traces. Due to the absolute value operator, the transform is no longer linear, and hence cannot be described in terms of Eqn. 2.11 with a suitable $\vec{a}$. In Sect. 2.5.4 and Sect. 2.5.5, we provide a comparison of our proposed technique to DFA.

It should be taken into account that computing the DFT over the complete trace is only suitable in special cases. In practice, a trace is usually split into windows of a given length w_{in}, which are processed separately. As of today, determining the optimal window length is a somewhat heuristic process that either involves (educated) guessing or optimization by testing many choices for the parameter.

Our proposed method can be combined with the frequency domain transformation. The weight vector is then applied to the transformed traces $|\mathrm{DFT}\{\vec{x}_i\}|$ and optimized according to Sect. 2.5.3. In cases where the leakage is distributed over multiple frequency bins, this approach can combine and thus presumably better utilize the overall side-channel information. Therefore, we also included this approach in our simulation and practical results in Sect. 2.5.4 and Sect. 2.5.5.

2.5.4 Simulation Results

In order to evaluate the effect of the optimized weight coefficients, we generated simulated traces for a 128-bit implementation of the AES in MATLAB. The main purpose of this section is to

demonstrate the basic effectiveness of the proposed approach. We do not aim to comprehensively examine every conceivable scenario, hence, the choice of the simulation parameters may appear somewhat arbitrary.

In our simulation, the clock frequency was set to $33.\overline{3}$ MHz, with the trace being sampled at 1 GHz. A clock cycle thus contains 30 samples. The i'th simulated trace for the clock cycle c was then generated as the sum of a "clock" signal multiplied by a leakage s_i^c and normally distributed noise as $\vec{x}_i^c = (1 + \sigma_{signal} \cdot s_i^c)\ \vec{t} + \vec{\mathcal{N}}(0, \sigma_{noise})$

We used $\sigma^2_{signal} = \sigma^2_{noise} = 1/1000$. $\vec{t}$ was set to a rectangular pulse, that is, $\vec{t} = (1, 1, \ldots, 1, 0, 0, \ldots, 0)$, with the first seven entries set to 1 (corresponding to $1/4$ of the full cycle) and the remaining 23 entries set to 0. To form the final simulated trace $\vec{x}_i$, we concatenated four cycles $\vec{x}_i^c$. The leakage in the first cycle s_i^0 was generated as the HW of the 128-bit AES state after the initial key addition and `SubBytes` operation. In the remaining three cycles, $s_i^{1/2/3}$ was calculated as the HW of a uniformly distributed random 128-bit value.

Band-Limited Noise

To simulate the effect of a band-limited noise source, we added an additional noise term $\vec{\mathcal{N}}_{band}$ to the simulated trace. For our experiments, we selected a noise bandwidth of ± 1 MHz around 24 MHz, i.e., the spectrum of $\vec{\mathcal{N}}_{band}$ is "white" between 23 and 25 MHz and zero otherwise. For a range of noise powers of $\vec{\mathcal{N}}_{band}$, we then performed (1) a time domain CPA, (2) a CPA on the frequency domain representation of the traces (cf. Sect. 2.5.3), (3) a time domain, and (4) a frequency domain CPA using optimized filter coefficients $\vec{a}$ as described in Sect. 2.5.3. For the profiling and the attack, we used different keys and different input data.

Figure 2.6 depicts the respective (maximum) correlation for the cases (1), (2), (3), and (4) for a noise power (i.e., the average standard deviation $\sigma_{bandlimited}$) of 1. The average signal power of a trace was $\sigma_{trace} = 0.56$, i.e., the given $\sigma_{bandlimited}$ correspond to a "Trace-to-Noise Ratio" (TNR) of approximately 0.5. Because the simulated traces already contain white noise, we avoid the term SNR here.

As evident in Fig. 2.6, the CPA using optimized filter coefficients (3) outperforms the normal CPA (1) and the frequency domain CPA (2) in the presence of band-limited noise. Table 2.1 summarizes the results, giving the absolute value of the correlation coefficient for the correct key after 50,000 traces (Tab. 2.1a) and the ratio between the correlation for the correct candidate and the maximum correlation for the wrong candidates (Tab. 2.1b). If this ratio is less than 1, the correct key can no longer be distinguished from the wrong candidates, i.e., the attack does not succeed.

With increasing $\sigma_{bandlimited}$ (i.e., decreasing TNR), the correlation coefficient for the correct key candidate is very close to or even below the correlations for the wrong key candidates using method (1), (2), or (4) after 50,000 traces. In contrast, the correlation for the correct key obtained with method (3) clearly exceeds the correlation for the wrong candidates after less than 5,000 traces in all cases. Computing the frequency response corresponding to the optimized coefficients, it turns out that the range from 23 to 25 MHz is attenuated, while the filter's transfer function is rather flat in the region of the clock frequency. The corresponding plot of the frequency response is given in Fig. 2.7a .

Interestingly, if no additional noise is present, method (3) provides worse distinguishability of the correct key than methods (1) and (2). In this case, the optimization algorithm appears

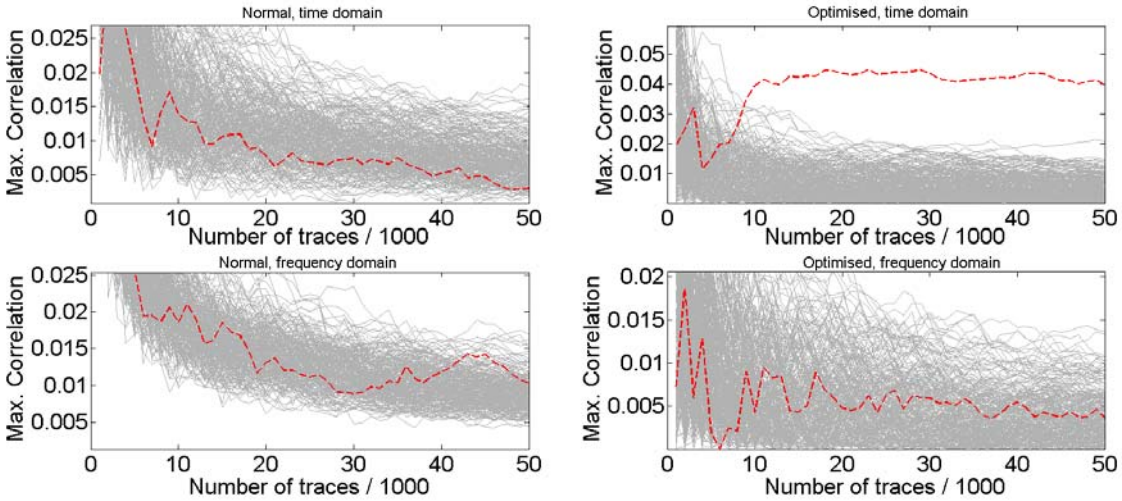


Figure 2.6: Maximum correlation for band-limited noise, $\sigma_{bandlimited} = 1$. Top left: time domain (1), top right: time domain, optimized (3), bottom left: frequency domain (2), bottom right: frequency domain, optimized (4). Correct key candidate: dashed, red

TNR	(1)	(2)	(3)	(4)
∞	**0.123**	0.09	0.051	0.017
1.7	0.009	0.019	**0.049**	0.014
1	0.005	0.012	**0.049**	0.007
0.5	0.003	0.01	**0.04**	0.004

(a) Correlation using 50k traces

TNR	(1)	(2)	(3)	(4)
∞	**3.73**	3.46	3	1.21
1.7	0.5	1	**2.88**	1.17
1	0.28	0.63	**2.72**	0.44
0.5	0.17	0.59	**1.9**	0.26

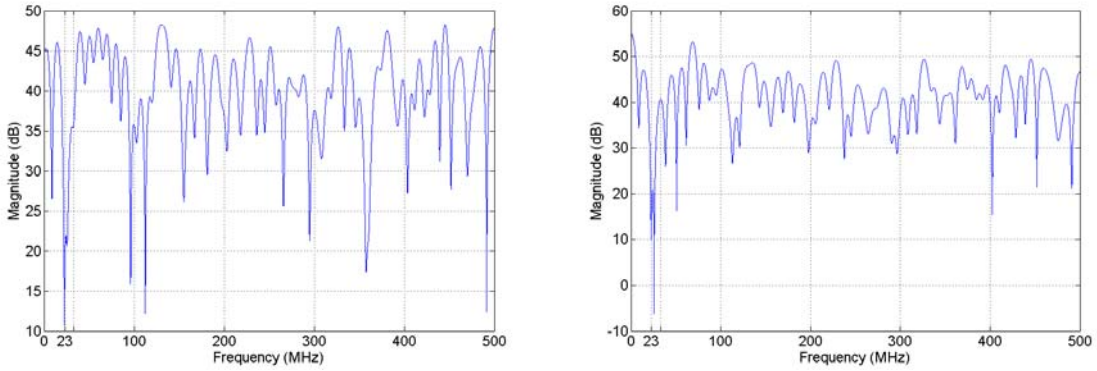
(b) Ratio between correct and maximum wrong candidate (50k traces)

Table 2.1: Comparison of evaluation methods (1) - (4) for simulated traces with band-limited noise. Best values in bold font

to overfit the weight coefficients—the coefficients then yield optimal distinguishability based on the data used in the profiling phase, but do not produce the desired effect in general. Thus, in the attack phase with a different set of traces, the distinguishability is reduced and not—as intended—increased. This problem could presumably be mitigated by techniques used in global optimization, e.g., running the optimization algorithm several times using different initial values and different subsets of the profiling data. For the purposes of this thesis, we did not look further into this issue and leave it for future work.

Timing Randomization

A randomization of the algorithmic timing was realized by shuffling the four clock cycles for each trace, that is, by randomly selecting a uniformly distributed position for the clock cycle that corresponds to the actual AES state. For this case, we applied the same evaluation methods as



(a) Noise ($\sigma_{bandlimited} = 1$)

(b) Noise + randomised timing (4 cycles)

Figure 2.7: Magnitude frequency response of the optimized filter coefficients for the simulated traces

above. We also combined the timing randomization with the band-limited noise source, again considering the same range of noise powers as for the non-randomized traces.

The result for $\sigma_{bandlimited} = 1$, i.e., a TNR of approximately 0.5, is exemplarily depicted in Fig. 2.8. Table 2.2 subsumes the results like in Tab. 2.1, giving the maximum correlation and the ratio between the correlation for the correct and the highest wrong candidate for different TNRs. While the normal CPA and the frequency domain CPA fail to clearly distinguish the correct key candidate from the wrong ones after 50,000 traces, the optimization approach determines filter coefficients that allow to extract the correct candidate after less than 15,000 traces. The according frequency response (Fig. 2.7b) again exhibits a band-stop characteristic eliminating the band-limited noise. In the time domain, the filter coefficients additionally resemble a comb filter, i.e., realize the averaging over multiple clock cycles.

TNR	(1)	(2)	(3)	(4)
∞	0.038	**0.089**	0.04	0.02
1.7	0.005	0.018	**0.04**	0.014
1	0.004	0.011	**0.038**	0.007
0.5	0.003	0.01	**0.029**	0.004

(a) Correlation using 50k traces

TNR	(1)	(2)	(3)	(4)
∞	1.9	**3.42**	2.5	1.33
1.7	0.28	1	**2.22**	1.08
0.5	0.22	0.58	**2.11**	0.54
1	0.17	0.59	**1.81**	0.25

(b) Ratio between correct and maximum wrong candidate (50k traces)

Table 2.2: Comparison of evaluation methods (1) - (4) for simulated traces with band-limited noise and timing randomisation (4 cycles). Best values in bold font

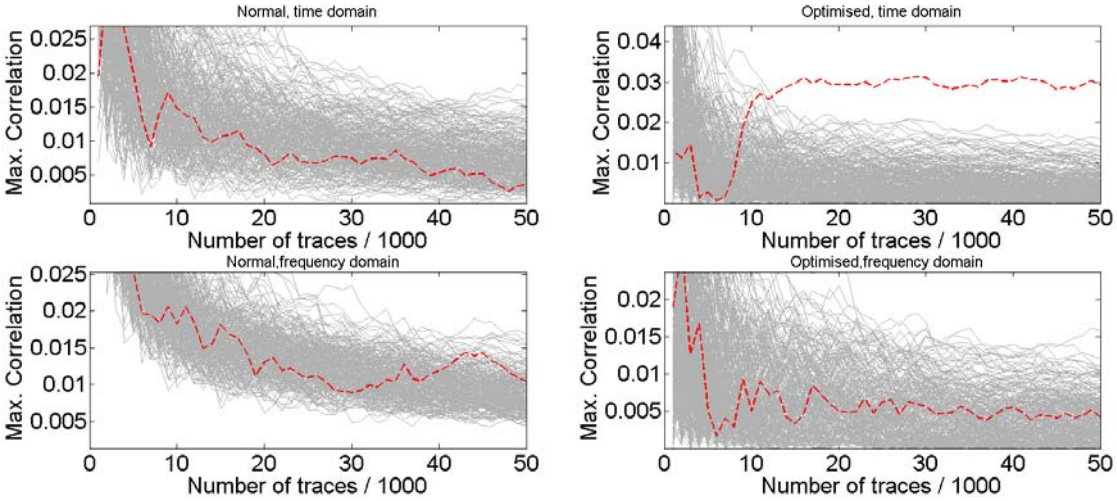


Figure 2.8: Maximum correlation for jitter (4 cycles) and band-limited noise, $\sigma_{bandlimited} = 1$. Top left: time domain (1), top right: time domain, optimized (3), bottom left: frequency domain (2), bottom right: frequency domain, optimized (4). Correct key candidate: dashed, red

2.5.5 Practical Results

In order to evaluate the efficiency of our findings in a real-world setting, we applied our methods to the traces provided in the second DPA contest [COM]. The traces were recorded for a hardware implementation of the AES on the Sasebo GII [Nat09] at a sample rate of $f_s = 5\,\text{GHz}$. We focused on the last round of the encryption process and accordingly only used the respective part from time point 2300 to 2700 of the 3253-point original traces.

For the profiling purposes, we used 15,000 raw traces of the "public database" `DPA_contest2_public_base_diff_vcc_a128_2009_12_23` belonging to the encryption with the AES key $k_{profiling}$ = `0x37d0d724d00a1248db0fead349f1c09b`. For the attack phase, i.e., to evaluate the effect of the optimized filter coefficients, we used 15,000 traces for k_{attack} = `0x00000000000000003243f6a8885a308d3`. These keys lead to different subkeys for the first S-box in the final round (`0xdc` for $k_{profiling}$, `0x53` for k_{attack}).

As a first step, we performed a standard CPA targeting the (byte-wise) HD between the input of the last `SubBytes` operation and the encryption result, following the reference attack of the DPA contest. As depicted in Fig. 2.9, the highest correlation coefficient clearly occurs for the correct key candidate after 15,000 traces, with a magnitude of 0.064 in the time domain and 0.055 in the frequency domain, respectively. However, significant "ghost peaks" occur at the end of the trace (around point 300).

At this point, we want to emphasize that we primarily use the DPA contest traces to demonstrate the general effectivity of our approach in a practical setting. Hence, we did not employ the full range of evaluation metrics provided by the contest. We applied the same evaluation methods as in Sect. 2.5.4, that is, CPA (1), frequency domain CPA (2), CPA with optimized coefficients (3), and frequency domain CPA with optimized coefficients (4).

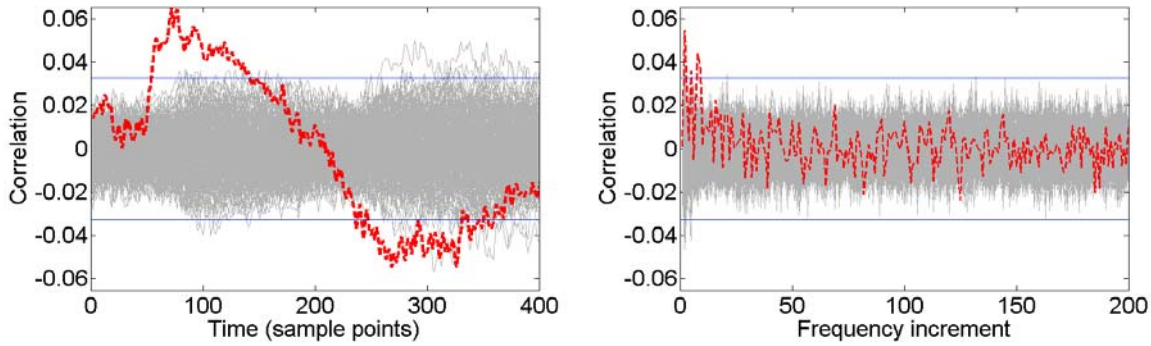


Figure 2.9: Correlation coefficients (DPA contest v2 AES) for the first byte after 15,000 traces, left: time domain, right: frequency domain. Correct key candidate: dashed, red

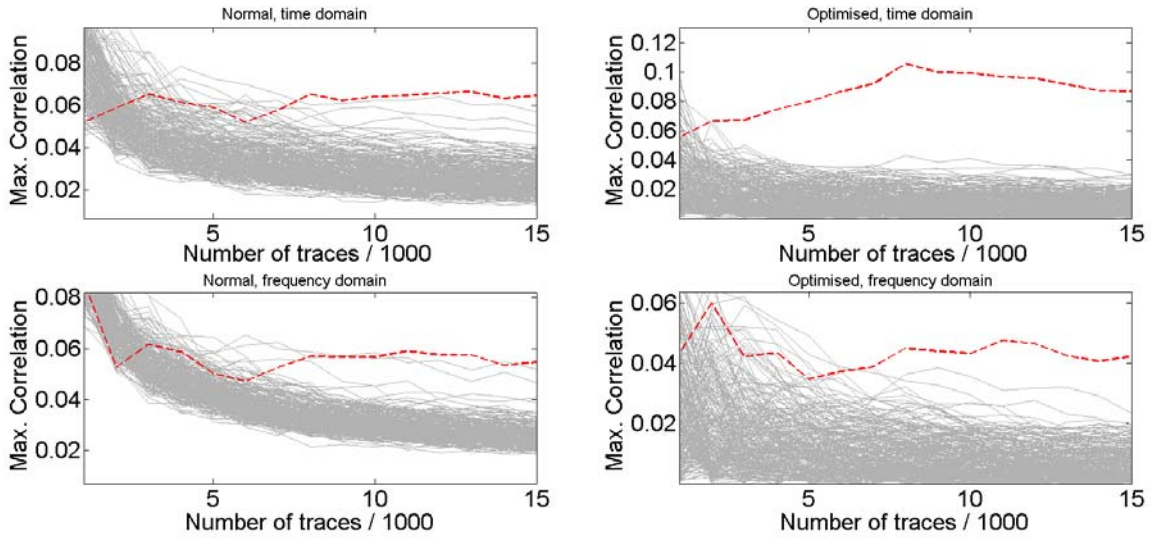
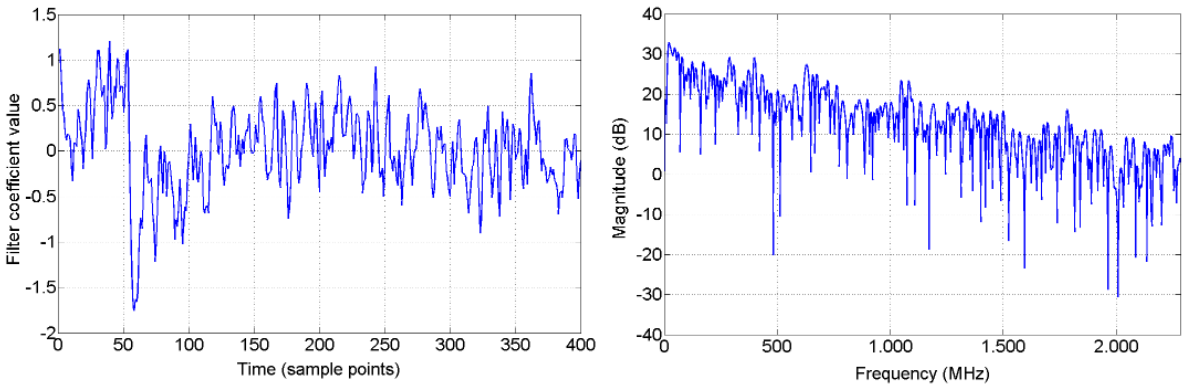


Figure 2.10: Maximum correlation coefficients (DPA contest v2 AES) for the first byte. Top left: time domain (1), top right: time domain, optimized (3), bottom left: frequency domain (2), bottom right: frequency domain, optimized (4). Correct key candidate: dashed, red

As evident in Fig. 2.10, the optimized coefficients decrease the number of required traces both for the time and the frequency domain CPA. For the normal CPA, the correct key candidate yields the highest correlation, however, the ratio with the second highest is rather small, i.e., $^{0.064}/_{0.057} = 1.12$ in the time domain and $^{0.055}/_{0.052} = 1.06$ in the frequency domain. In contrast, with the optimized filter coefficients, these ratios are increased to $^{0.087}/_{0.03} = 2.9$ and $^{0.042}/_{0.023} = 1.83$, respectively. Accordingly, the (approximate) minimum number of traces needed to distinguish the correct and the wrong key candidate is reduced from 8,000 to 3,000 in the time domain and 11,000 to 8,000 in the frequency domain.

The optimized coefficients reduce the influence of the "ghost peaks" mentioned above on the results of the CPA, i.e., improve the distinguishability of the correct key candidate. As evident



(a) Time domain

(b) Frequency response for (a)

Figure 2.11: Optimized filter coefficients for the DPA contest v2 traces

in Fig. 2.11a, the optimized coefficients obtained with method (3) put the highest weight on the maximum of the leakage at around time point 50, followed by decaying weight according to the shape of the correlation depicted in Fig. 2.9. The according frequency response (Fig. 2.11b) shows a lowpass characteristic in general. However, certain frequencies are selectively attenuated, for example, narrow regions around 70 MHz, 160 MHz, 227 MHz, 327 MHz, 388 MHz, 422 MHz, and 480 MHz.

We experimentally verified that these results obtained for the first byte equivalently hold for the other bytes. Table 2.3 gives the ratio between the correlation for the correct and the highest wrong candidate after 15,000 traces for a normal CPA (1) and a CPA with optimized coefficients (3) in the time domain. For all bytes, the optimized coefficients lead to a higher distinguishability and allow for extracting the key with less traces.

Byte	1	2	3	4	5	6	7	8
CPA	1.12	1.84	1.47	1.67	1.02	1.77	1.5	1.45
opt. CPA	2.9	2.74	2.48	3.83	1.71	3.88	3.12	3.15
Byte	9	10	11	12	13	14	15	16
CPA	1.68	2.17	1.24	1.83	1.63	1.68	1.53	1.81
opt. CPA	3.44	3.79	2.16	3.31	2.49	3.24	2.54	4.44

Table 2.3: Ratio between correct and maximum wrong candidate for normal (1) and optimized CPA (3) in the time domain after 15,000 DPA contest v2 traces

2.6 Conclusion

In this chapter, we introduced the basics of SCA as used in this thesis, covering both various measurement (power, EM, and so on) and evaluation methods (SPA, DPA/CPA, and TA). These techniques form the foundation of the real-world attacks presented in Chap. 7–11.

In addition, we presented a closed form to perform CPA on traces under a linear transform. In contrast to traditional approaches, our method does not require to re-compute the CPA for each particular choice of the transform parameters. Thus, in cases where many different parameters are to be tested, our method allows for a substantially faster evaluation. Consequently, we derived an optimization criterion allowing to find an "optimal" transform in the sense that it maximizes the distinguishability of the correct key candidate. Using both simulated and real-world traces, we demonstrated that this technique performs better than traditional methods and offers a systematic way to derive linear filters for SCA.

Note that this technique was developed *after* performing several practical side-channel attacks with heuristically selected pre-processing parameters. Hence, we did not use the automatic, optimization-based approach in our real-word analyses. We rather consider this method as a starting point for further research with respect to SCA pre-processing. However, right now, especially when designing countermeasures, our method allows to give a more comprehensive (and objective) assessment regarding the effectiveness of protection mechanisms than the heuristic selection of one particular set of parameters.

Part II

Tools

Chapter 3

Measurement and Evaluation

For performing implementation attacks in practice, a range of hardware and software tools is required. First, especially for DUTs with an RF interface, suitable means of communicating with the DUT are needed. Second, for side-channel attacks, a suitable DSO with additional equipment like EM probes and amplifiers is necessary. The measurement process is controlled by software running on a PC. Finally, the acquired measurements have to be processed and statistically evaluated for an SCA. In this chapter, we describe the developed toolchain to carry out real-world implementation attacks, detailing on the different components of our framework.

Contents of this Chapter

3.1 Receiving and Transmitting RF Signals

Many modern (cryptographic) devices no longer employ a contact-based interface but use a wireless link instead. For instance, contactless RFID smartcards (cf. Chap. 6 and Chap. 7) have become quasi-standard for mobile payments, public transport systems, access control, and governmental ID documents. Since many standards and frequencies are used for different target devices, a flexible way of communicating with such wireless devices is needed. In this section, we describe the two main tools used for this purpose throughout this thesis, the Universal Software Radio Peripheral (version 2) (USRP2) (Sect. 3.1.1) and a custom RFID reader (Sect. 3.1.2).

3.1.1 The USRP2

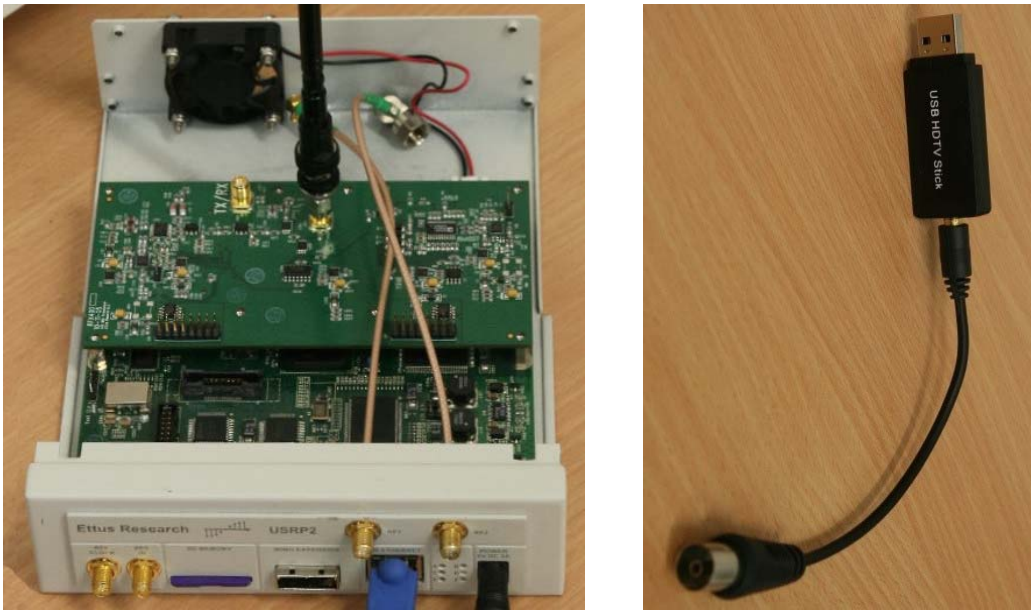
The USRP2 is an SDR device designed by Ettus Research [Ett13b] for transmitting and receiving RF signals over a wide frequency range and with arbitrary modulation schemes: a signal to be transmitted is specified as a series of (baseband) I/Q samples that are converted to analog signals by two 16-bit Digital-Analog Converters (DACs) and then up-converted to the desired RF frequency and further amplified by so-called daughterboards.

These daughterboards (of which two can be installed in the USRP2) perform the necessary analog processing for a specific frequency range and are available for, e.g., lower frequencies

(1–250 MHz), specific bands (like the Industrial, Scientific, and Medical (ISM) frequencies at 433 MHz, 868 MHz, and 2.4 GHz), and also very wide ranges (50 MHz–2.2 GHz or 400 MHz–4.4 GHz) [Ett13a].

Similar to transmitting signals, the daughterboards allow to receive signals, down-converting them from the RF frequency to the baseband. The USRP2 then digitizes the I/Q components using two 14-bit Analog to Digital Converters (ADCs). The raw baseband signals are then available to the controlling PC (connected via a Gigabit Ethernet link) and can be further digitally processed.

The SDR approach of the USRP2 allows for up to 25 MHz of RF bandwidth, which, in the case of the DUTs considered in this thesis, is rarely needed, since the devices usually exchange a small amount of data at rates of a few hundred $^{\text{kBit}}/_{\text{s}}$ at best. For controlling the USRP2, we used the `GNURadio` [GNU13] software, which allows to develop SDR applications using a network of interconnected blocks. In some cases, we additionally employed the low-level drivers provided by Ettus research to directly retrieve or generate signals from custom C++ programs.



(a) USRP2 (b) RTL2832U USB stick

Figure 3.1: The SDRs employed in this thesis (photos taken by Thomas Rudolph)

The USRP2 (case opened, RFX400 transmitter daughterboard installed) is depicted in Fig. 3.1a. Occasionally, we also used a—compared to the USRP2—much simpler, RTL2832U-based [Rea13] USB stick depicted in Fig. 3.1b. This device was originally designed as a Digital Video Broadcasting – Terrestrial (DVB-T) receiver, however, can with the software `rtl-sdr` [Osm13] function as a receive-only SDR for frequencies from approximately 50 MHz to 2.2 GHz with a maximum bandwidth of 1.6 MHz. In terms of sensitivity and flexibility, the USRP2 has by far better characteristics, however, in scenarios where the sensitivity requirements are moderate and where a small, mobile SDR is helpful, using the DVB-T stick is advantageous.

Besides, the device has a cost of approximately USD 20, compared to approximately USD 2,000 for the USRP2 with basic daughterboards.

In this thesis, the USRP2 is used in Chap. 6 as one of the main tools to determine the maximum achievable eavesdropping and active communication ranges for both active Ultra High Frequency (UHF) and passive High Frequency (HF) RFID transponders. Besides, for Chap. 8, the USRP2 was used to analyze and reverse-engineer the RF protocol of a proprietary access control system.

3.1.2 Custom RFID Reader

A custom, freely programmable RFID reader developed by Kasper et al. [KCP07] was employed for the communication to RFIDs operating at 13.56 MHz. The device is compliant to ISO 14443 [Int01b] and ISO 15693 [Int09] and has a cost of approximately USD 50 (for the components and the PCB) . The schematics and the software for the reader are available under an open-source license on Sourceforge [Sou13b]. The device (without the antenna being connected) is depicted in Fig. 3.2.

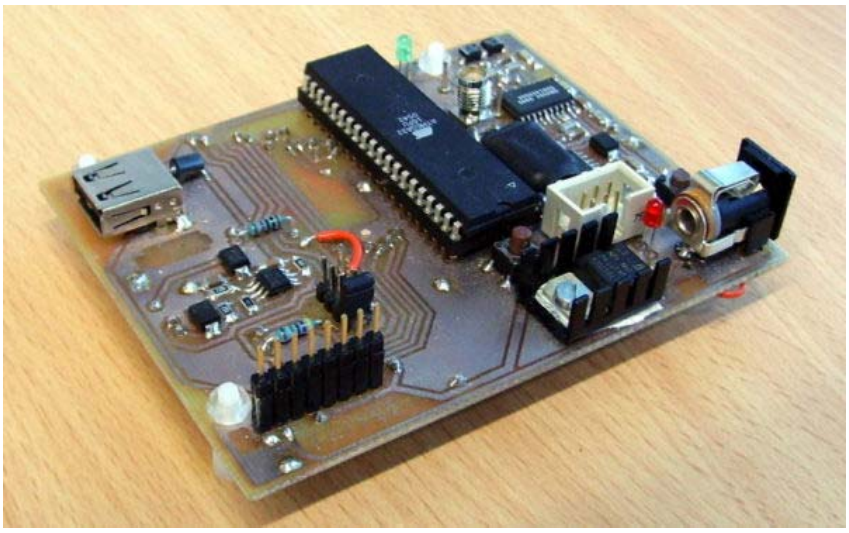


Figure 3.2: Custom reader for HF RFIDs, no antenna connected. Left: USB connector, right: antenna and DC power connector

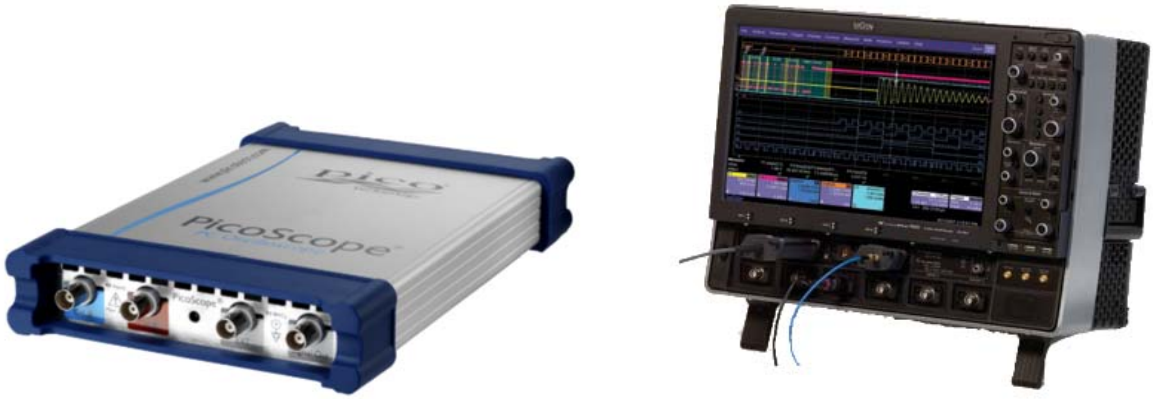
In contrast to commercially available readers, this device gives control over the transmitted messages down to the bit level and thus, e.g., allows to implement protocols of arbitrary contactless smartcards that conform to the respective ISO standards. Besides, in the context of SCA, the reader can be programmed to generate a trigger signal on a General Purpose I/O (GPIO) pin when the command starting a cryptographic operation on the RFID transponder is sent.

In Chap. 7, the custom RFID reader is employed to communicate with the Mifare DESFire MF3ICD40 smartcard to acquire traces for an SCA. For the analysis of the real-world public transport card “Opencard” in Sect. 7.5, the reader is used to read the data on the card after the secret keys have been extracted.

3.2 Side-Channel Measurement

3.2.1 Oscilloscopes

For acquiring side-channel measurements in particular and for analyzing and developing analog and digital circuits in general, we utilized two DSOs: the Picoscope 5204 [Pic08] and a LeCroy WavePro 715Zi [Tel13]. The devices are depicted in Fig. 3.3a and Fig. 3.3b.



(a) Picoscope 5204 (b) LeCroy WavePro 715Zi

Figure 3.3: The DSOs employed in this thesis

The LeCroy WavePro 715Zi is a standalone oscilloscope with a built-in PC and a cost of approximately USD 25,000. It features a bandwidth of 1.5 GHz (for 50 Ohm input impedance), a maximum sample rate of 20 GHz, and four input channels. The DSO has a 128 MSample waveform memory and a minimum sensitivity of 1 $^{\mathrm{mV}}/_{\mathrm{div}}$ (for 1 MΩ input), which translates to an input range of ± 4 mV (eight vertical divisions).

In contrast, the Picoscope 5204 is a dual-channel DSO that needs to be connected to a PC via USB 2.0 to be operated. The devices features a maximum sample rate of 1 GHz, an 8-bit vertical resolution, and a 128 MSample waveform memory. The input bandwidth is 250 MHz (for 10:1 probes), with a minimum input range of ± 100 mV. Additionally, the device offers a separate external trigger input and a built-in arbitrary signal generator.

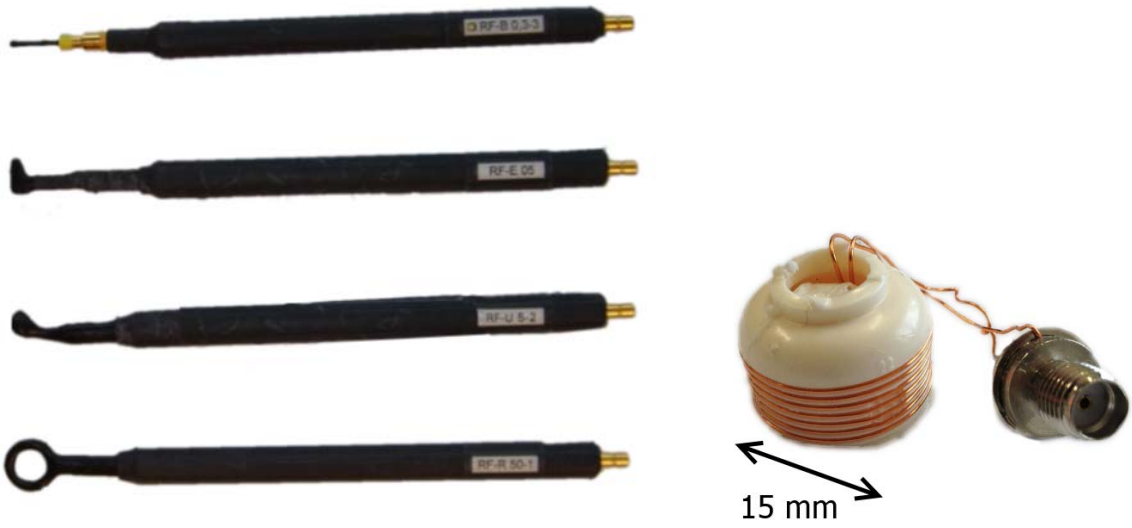
For most side-channel attacks, it turned out that the Picoscope 5204 had a sufficiently high sample rate and sensitivity: only the SCA of the Altera Stratix II bitstream encryption described in Chap. 11 was carried out with the LeCroy WavePro 715Zi. Even in this case, this DSO was mainly used because it had already been used for the initial profiling—likely, the attack could also be done with the Picoscope.

The fact that the Picoscope 5204 is often sufficient underlines the low-cost nature of our attacks: the Picoscope can be bought for approximately USD 3,000, i.e., for less than one eighth of the price for the WavePro 715Zi. Going further, a version with 32 MSample memory, the Picoscope 5203, is available for approximately USD 1,100, i.e., for approximately $^{1}/_{23}$ of the WavePro's price. Even a DSO with this (relatively small) amount of memory would suffice for the attacks demonstrated in this thesis.

3.2.2 Amplifiers and EM Probes

For SCA, signals with often small amplitudes in the range of a few mV have to be digitized. Since the Picoscope 5204 has a relatively large minimum input range of $\pm 100\,\text{mV}$, we employed an external pre-amplifier when necessary. To this end, we used the commercially available amplifier PA303 made by Langer EMV [Lan]. The device has an almost constant gain of 30 dB (i.e., a multiplication by a factor of 31.6) up to a frequency of 3 GHz. The amplifier's output amplitude is, according to information by the vendor, limited to $\approx 1\,\text{V}$. For higher voltages, non-linear effects like clipping may occur. Thus, the maximum input amplitude is approximately 31 mV.

When capturing the EM emanations of a DUT, e.g., for the DESFire MF3ICD40 in Chap. 7, the Yubikey 2 in Chap. 5, or the SimonsVoss door lock in Chap. 8, we used the near-field EM probes made by Langer EMV that are depicted in Fig. 3.4a. The set of four probes has a cost of approximately USD 300.



(a) Langer EMV, top to bottom: RF-B 0,3-3, RF-E 05, RF-U 5-2, RF-R 50-1

(b) Self-made probe

Figure 3.4: EM probes: Commercial set of probes by Langer EMV and custom probe made using copper wire

Most often, we used the (magnetic) probe termed "RF-U 5-2" (third one from the top in Fig. 3.4a). Alternatively, we also experimented with probes made of a few turns of standard wire. Figure 3.4b shows such a probe, built with seven turns of copper wire on an audio connector casing (diameter 15 mm) made of plastic, with the wire soldered to a standard SubMiniature version A (SMA) jack. Such a simple EM probe can be built for a few USD and showed, at least in experiments with the DESFire MF3ICD40 (Chap. 7), a similar performance to the much more expensive commercial EM probes[1].

[1] Which are of course easier to handle, smaller, and more versatile

3.2.3 Measurement Framework

The measurement software component described in this section (and the evaluation part in Sect. 3.3) is a partially extended and improved version of the framework developed in [Osw09]. Hence, we here only summarize the most important aspects of the framework, while further details can be found in [Osw09]. The typical setup for SCA measurements is depicted in Fig. 3.5. The controlling PC communicates with the DUT, for instance, to trigger a cryptographic operation and to retrieve the result of the computation on the DUT. At the same time, the DSO (e.g., the Picoscope 5204 or the WavePro 715Zi) records a side-channel trace (e.g., of the EM emanation or the power consumption) that is read and stored by the PC.

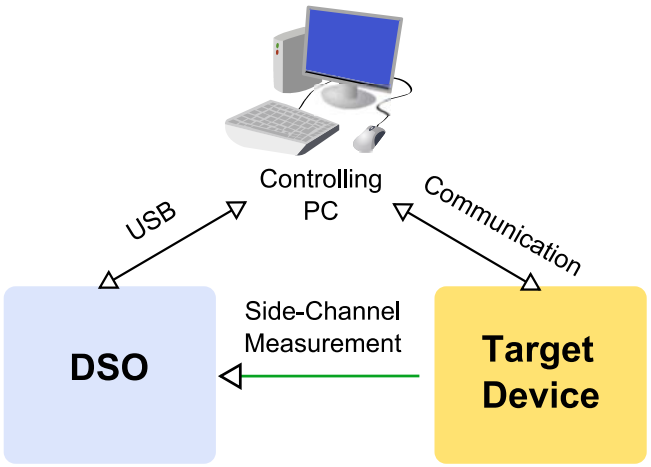


Figure 3.5: Typical measurement setup for SCA: The controlling PC communicates with the target DUT and acquires the side-channel measurements using a connected DSO

Note that in the case of the Picoscope 5204, a separate PC is used, while for the WavePro 715Zi, the built-in PC could perform the control tasks. The recorded trace is stored and the measurement process repeated until the desired amount of traces has been acquired. Clearly, having a re-usable measurement software which hides the details of, e.g., the driver to communicate with the DSO, is desirable. In our framework, the Application Programming Interface (API) is kept as abstract as possible so that different DSOs can be used, e.g., when higher precision or bandwidth is needed.

Both the measurement and the evaluation part of our framework are controlled by configuration files to keep the parameters separate from the program code. Each experiment (i.e., set of traces and related data) is stored in a separate folder together with the relevant configuration files. This approach enables to reproduce experiments, facilitates the evaluation of the traces, and allows for automatic batch measurement series and processing of the results.

The configuration files are standard `.ini` files and thus have a key-value pair syntax. Separate sections in the file control the different sub-modules, e.g., the measurement and trigger settings of the DSOs and the communication with the DUT. For more information, cf. Chap. 3 in [Osw09]. The following example illustrates the basic syntax of a configuration file, showing the section that controls the DSO settings:

```
# ... other entries ...
[scope]
# Which scope is used: picoscope (default), picoscope_compressed (8-bit),
#                      adc8, adc16, adc32
type = picoscope_compressed
# Sample rate in Hz
sample_rate = 500000000
# ... other entries ...
```

The begin of a section is indicated by the name in square brackets, followed by a number of key-value pairs (separated by `=`). The `#` character marks a comment, with all following characters on the line being ignored. To read the configuration files, the *iniParser* library [Dev09] is used, which provides functions to obtain the value given a specific key in a section.

Most functionality of a measurement application written for a specific DUT is provided and defined in the C++ class `measurement_app`. We followed a common approach of object-oriented frameworks and developed a base class with methods to be overwritten in derived classes to set the behavior for a specific task. The class encapsulates commonly needed functionality while maintaining full flexibility, i.e., allows the user to modify these pre-defined functions if necessary. Figure 3.6 shows the control flow of a standard measurement application using `measurement_app`. The highlighted methods are to be overwritten in order to customize the framework for a concrete DUT.

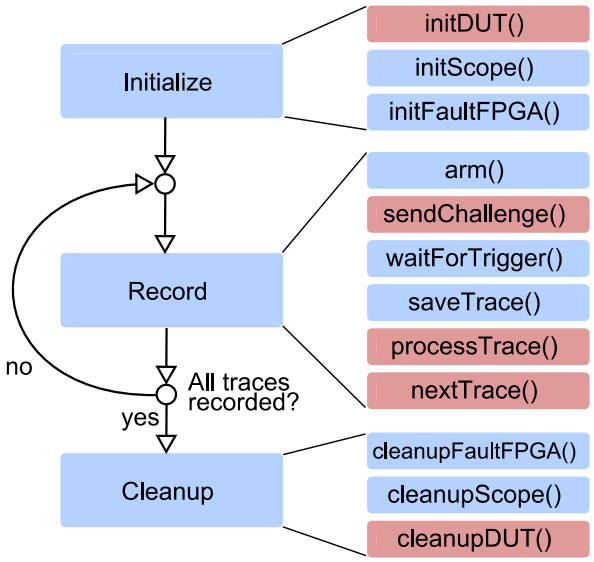


Figure 3.6: Program flow of a measurement framework application using the class `measurement_app`

The methods to modify mainly implement the communication with the DUT: `initDUT()` sets up a connection that is closed by `cleanupDUT()` at the end. The challenge for the DUT is sent in `sendChallenge()` and updated in `nextTrace()`. Optionally, the recorded waveform can be processed in `processTrace()`, e.g., to remove traces that contain distortion already during

the measurement process. Note that FI is taken into account in the measurement process. For instance, one may choose the next challenge or discard traces based on the outcome of a prior FI.

3.3 Evaluation Framework

For evaluating side-channel traces, e.g., in a CPA or TA, cf. Sect. 2.3, and for DSP pre-processing, we extended the framework of [Osw09]. Analogous to the measurement framework, we developed, amongst others, the C++ class `processing_app` as the basis for processing and evaluating the recorded traces. The approach is similar to `measurement_app`, i.e., methods are overwritten to implement the desired, case-specific functionality, while standard functions are provided by the base class. These include loading the trace files in the correct order, digital pre-processing methods (e.g., filtering, Fast Fourier Transform (FFT) computations, etc.), and the alignment in time using pattern matching methods.

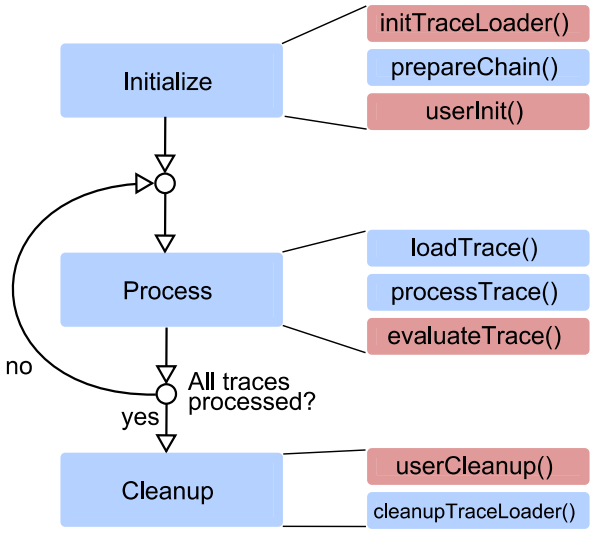


Figure 3.7: Program flow of an evaluation framework application using the class `processing_app`

The program structure is depicted in Fig. 3.7, marking the methods commonly changed for a specific problem. `initTraceLoader()` lets the application select a specific trace file format. The framework supports the straightforward 8 or 16-bit format of the Picoscope, `double` values (stored in 8 byte), and the format of the LeCroy WavePro 715Zi, cf. Sect. 3.2.1. Due to the modular design, other formats can be added as well. `userInit()` then initializes the actual functionality of the program, e.g., prepares a CPA. The respective cleanup function called before the application quits is `userCleanup()`. `evaluateTrace()` is called for each loaded and pre-processed trace and may, for instance, add the trace to a CPA, compute the average, and so on.

During the evaluation, each trace is digitally pre-processed in a processing chain before being passed to `evaluateTrace()` , i.e., the signal is modified by a sequence of filtering blocks.

The processing chain is configured in a configuration file and provides blocks for extracting parts of a trace, rectification, digital highpass and lowpass filters, pattern matching functionality for alignment purposes, the computation of the power spectrum (i.e., the squared magnitude of the Fourier transform), and for performing the Dynamic Time Warping (DTW) algorithm [vWWB11].

The evaluation of the pre-processed signal then depends on the concrete target. We developed applications for, amongst other, performing a CPA on 3DES and AES, in both cases including a mode that tests numerous intermediate values and power models (for a known key) in the profiling phase. These applications were for instance employed for the attacks on the DESFire MF3ICD40 in Chap. 7, the Yubikey 2 in Chap. 9, and the Altera Stratix II FPGAs in Chap. 11. Besides, we built an application to compute the mean traces and covariance matrices needed for a TA, cf. Sect. 2.3.

The developed C++ framework was most often used when a larger number of traces and many key candidates were involved. Thus, for the SCA of the SimonsVoss access control system (Chap. 8) and the Maxim SHA-1 ICs (Chap. 10) where in both cases a maximum of a few thousand traces was needed, we used scripts developed in `MATLAB` due to the significantly faster and easier development process. In general, `MATLAB` was employed as a data visualization and processing tool in many parts of the thesis.

3.4 Conclusion

The equipment and software components presented in this chapter form the basis for our attacks on real-world devices. Interestingly, it turned out that in many cases low-cost alternatives are available for expensive measurement devices. The USRP2 can sometimes be substituted with simple DVB-T sticks, a custom RFID reader can be employed, an EM probe be made using copper wire, or an expensive standalone DSO be replaced with the USB-based Picoscope.

For controlling the measurement process, the C++ framework of [Osw09] was extended and subsequently utilized in various practical analyses. Similarly, the evaluation tools (written in C++ and `MATLAB`) were improved and adapted to specific attack scenarios, which was possible due to the framework's modular design. The tools described in this chapter were subsequently employed for carrying out the real-world attacks presented in Chap. 6–11. During this process, several additional improvements became necessary due to practical experiences, a fact that helped to enhance the framework further.

Chapter 4

Pre-Processing for SCA of RFID

Capturing the side-channel leakage of contactless RFID smartcards is problematic due to the high amplitude of the field generated by the reader. In this chapter, we present two designs of analog demodulators specifically made for refining the SCA of contactless smartcards. The customized analog hardware increases the quality of EM measurements and can be easily integrated into the existing SCA setup. Employing our developed demodulator to obtain power profiles of several real-world cryptographic RFIDs, we demonstrate the effectiveness of our measurement approach that forms the basis for the attack on the Mifare DESFire MF3ICD40 in Chap. 7.

Contents of this Chapter

4.1 Introduction

For the SCA of cryptographic RFIDs and contactless smartcards in particular, we built an optimized measurement setup. As explained in Sect. 2.2.3, analog demodulation can be used to separate the actual power consumption signal from the carrier signal and to thereby improve the quality of the (exploitable) side-channel leakage. To this end, we developed circuits to perform the analog demodulation theoretically described in Sect. 2.2.3. Figure 4.1 gives an overview over the main components.

The RFID reader described in Sect. 3.1.2 supplies the contactless smartcard with power and handles the communication, for instance, to trigger an encryption operation. An EM probe with the suitable pre-amplifier (Sect. 3.2.2) captures the magnetic near-field in the proximity of the IC, resulting in a "raw" signal (denoted as ② in Fig. 4.1) which is dominated by the 13.56 MHz carrier frequency of the reader.

On the one hand, this signal is directly recorded and stored using the Picoscope 5204 DSO (cf. Sect. 3.2.1), on the other hand, it is passed to an analog demodulator that performs the operations outlined in Sect. 2.2.3 to facilitate SCA, resulting in the signal ① in Fig. 4.1. The

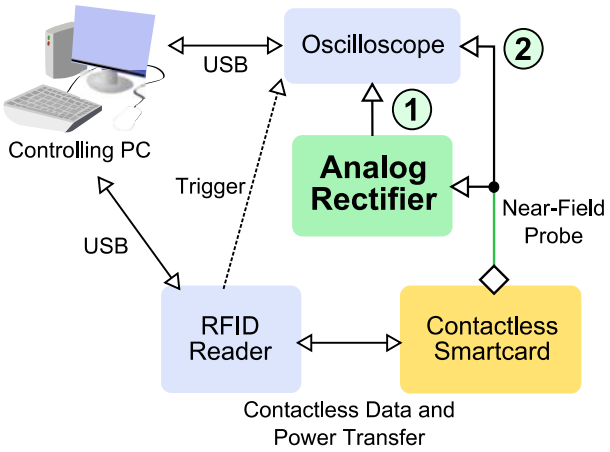


Figure 4.1: Overview over the measurement setup for SCA of RFIDs

central PC controls the measurement process, i.e., prepares and sends commands to the DUT via the RFID reader and acquires and stores the resulting side-channel traces ① and ②.

4.1.1 Contribution

In this chapter, we detail on the implementation of two demodulators—a half-wave rectifier and a full-wave rectifier—in Sect. 4.2 and Sect. 4.3, respectively. In Sect. 4.4, we quantify the influence of additional digital pre-processing. Finally, we give examples for traces of various contactless smartcards recorded with our setup in Sect. 4.5 and conclude in Sect. 4.6.

The tools described in this chapter were presented at RFIDSec 2011 [KOP12] and used for the analysis of the Mifare DESFire MF3ICD40 (cf. Chap. 7) published at CHES 2011 [OP11]. Part of this work was done jointly with Timo Kasper.

4.2 Half-Wave Rectifier

Initially, to minimize the complexity of the PCB, we implemented half-wave rectification, i.e., discarded the negative part of the signal. As explained in Sect. 2.2.3, this approach limits the bandwidth of the receivable signal to $^{13.56}/_{2}$ MHz. We designed a custom PCB comprising circuitry for amplification, rectification, and filtering of the raw analog signal.

The signal acquired by the EM probe is first amplified using an AD8058 Operational Amplifier (OP) [Ana09a] and then rectified by a BAT48 Schottky diode [Visb]. The result is filtered using an active low-pass filter. The filter is built with a Sallen-Key topology using an AD8045 OP [Ana04] with a -3 dB frequency of 6.25 MHz. The output stage is designed to drive a standard 50 Ω load, e.g., connected via a suitable coaxial cable. The overall structure of the demodulation system is depicted in Fig. 4.2.

In our practical experiments, it turned out that the filtering and amplification performed on the demodulation board can be further improved: the carrier signal was not suppressed as strong as desired and a slight drift of the DC component of the signal occurred during long-term measurements (e.g., due to temperature variations). Thus, we extended the demodulator with a bandpass filter circuit that further reduces the amplitude of the 13.56 MHz signal and besides

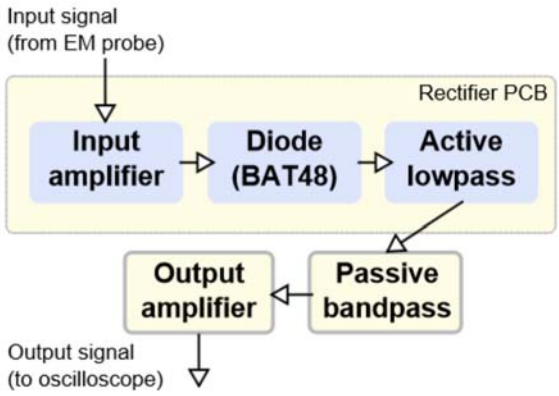


Figure 4.2: Structure of the half-wave rectifier circuitry

provides a highpass characteristic to remove the DC shift. This second-order filter was built using passive components only (i.e., inductors and capacitors) and has a passband between 100 kHz and 7 MHz. Finally, the filter output is again amplified with an OP to utilize the full input range of ± 100 mV of the Picoscope 5204 (cf. Sect. 3.2.1).

4.3 Full-Wave Rectifier

In extension of the half-wave rectifier, we developed a custom PCB comprising a full-wave rectifier and appropriate filter circuitry to perform the incoherent demodulation approach. Figure 4.3 shows the basic structure of the demodulation circuitry, while Fig. 4.4 gives the respective schematics.

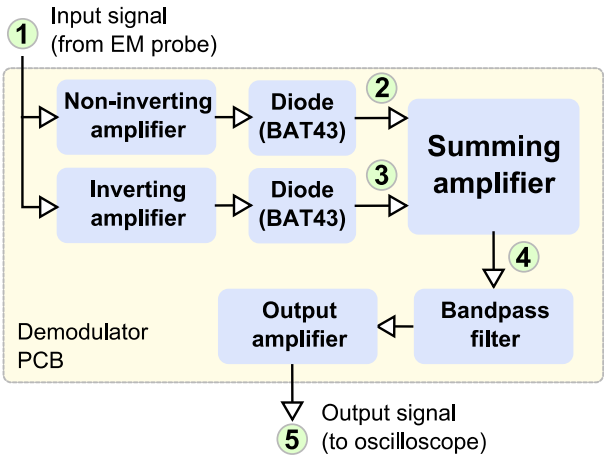


Figure 4.3: Structure of the full-wave rectifier circuitry

The full-wave rectifier is formed by two isolated half-wave rectifiers, each employing an BAT43 Schottky diode [Visa]. To rectify the negative part of the input ①, the signal is first inverted

and then rectified by the diode, yielding signal ③ in Fig. 4.3. For the positive portion, the buffer amplifier only provides isolation of the input signal and driving of the corresponding diode, but does not perform inversion to produce signal ②. The two resulting parts ② and ③ are then added to form the full-wave rectified output ④.

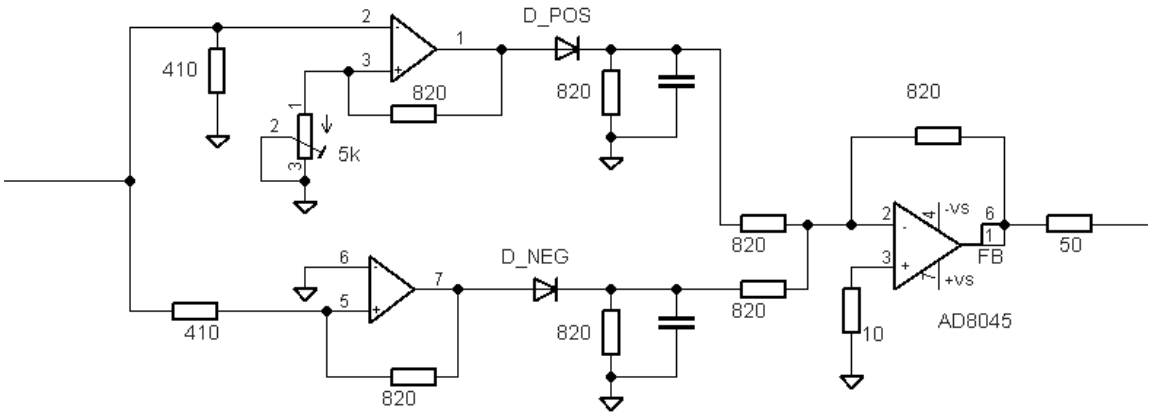


Figure 4.4: Schematics of the analog full-wave rectifier circuit

A third-order LC bandpass filter extracts the baseband part, i.e., the portion of the spectrum centered around 0 Hz. In our case, the −3 dB frequency was specified to 12 Mhz. Additionally, the filter also suppresses frequency components below 10 kHz to remove the constant part of the modulating signal. Finally, the output amplifier adjusts the amplitude of the signal in order to optimally utilize the minimum input range of ±100 mV of the Picoscope 5204 and drives a 50 Ω load. The schematics of the filter are given in Fig. 4.5.

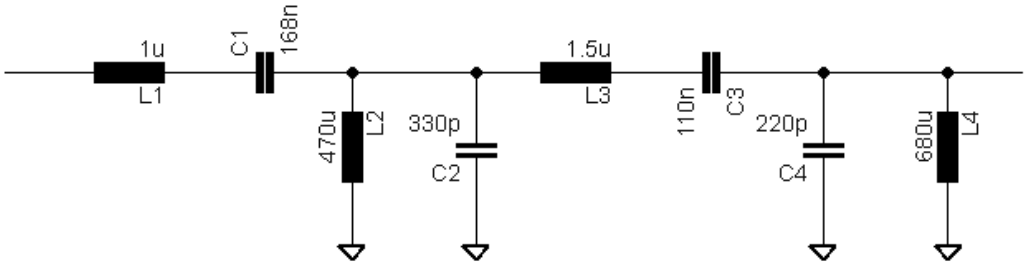
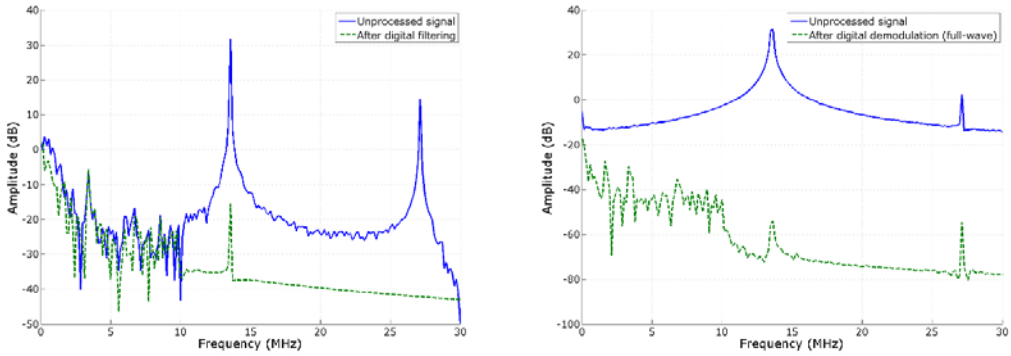


Figure 4.5: Schematics of the analog bandpass filter

4.4 Digital Processing

In the case that a raw signal (i.e., ② in Fig. 4.1) is used for SCA, it was shown in [KOP09] that the demodulation has to be performed digitally in order to conduct a successful CPA, i.e., digital pre-processing is mandatory. For the output of the analog demodulator, digitally



(a) Output of analog rectifier

(b) Raw signal before analog rectifier

Figure 4.6: Power spectrum before (solid, blue) and after (dashed, green) digital processing

filtering the output signal is optional, however, might help to further reduce the 13.56 MHz frequency component still present due to certain characteristics of the analog circuits.

In order to illustrate the effect of the analog and digital processing in summary, Fig. 4.6 shows the power spectra[1] of signals before (Fig. 4.6a) and after (Fig. 4.6b) digital processing, respectively. These results were obtained for the measurement of the SCA leakage of the Mifare DESFire MF3ICD40 with the full-wave rectifier, cf. Chap. 7 for a detailed description.

4.5 Measurements for Various Contactless Smartcards

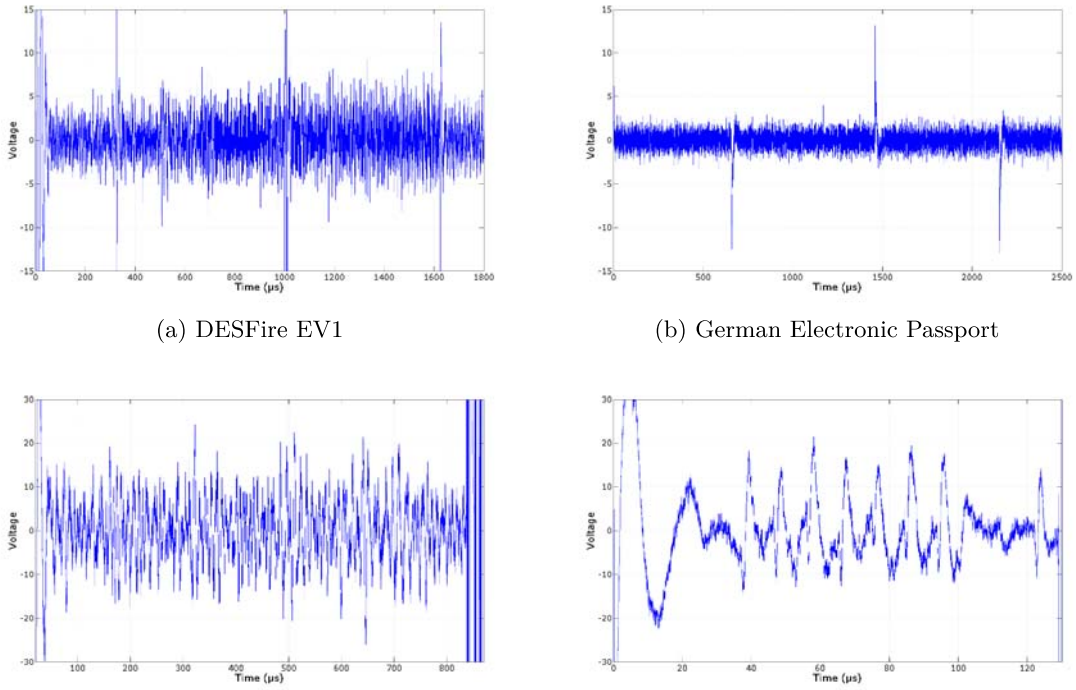
In order to further illustrate the capabilities of our setup and to show the general validity of the RFID leakage model of Sect. 2.2.3, we analyzed several contactless smartcards. As our main result, in Chap. 7 we present a successful side-channel attack on the Mifare DESFire MF3IC40 using the full-wave rectifier.

The selection in this section primarily focuses on modern high-security ICs (including the new DESFire EV1 and the German electronic passport), but also comprises devices for low-cost applications, i.e., Mifare Classic and Ultralight C. Note that the results are not specific to RFIDs devices manufactured by NXP—in fact, we were able to carry out successful SCA attacks on products made by other vendors, but cannot disclose the results for legal reasons.

In this section, we do not perform a detailed analysis of the considered DUTs as done for the Mifare DESFire MF3ICD40 in Chap. 7, but summarize the characteristics of the respective smartcards and provide some observations made during our experiments.

Figure 4.7 depicts exemplary power profiles of the DUTs, recorded with the half-wave rectifier during a particular cryptographic operation. The following paragraphs introduce the devices and give a short description of the cryptographic operations for which the side-channel traces were obtained. We highlight some particular features evident in the power profiles and try to relate them to the internal operation of the DUT.

[1] i.e., the squared magnitude $|\mathrm{DFT}\{s(t)\}|^2$ of the DFT



(a) DESFire EV1

(b) German Electronic Passport

(c) Ultralight C

(d) Mifare Classic

Figure 4.7: Power profiles of various contactless smartcards

Mifare DESFire EV1

The Mifare DESFire EV1 [NXP10b] is the successor of the Mifare DESFire MF3ICD40 and was announced in 2006. Apart from authentication and encryption with 3DES, the smartcard also provides support for AES with a 128-bit key. The device is certified according to Common Criteria (CC) EAL-4+ [BSI08] and implements special hardware countermeasures against SCA, which are, however, not further characterized in the publicly available datasheets. The authentication protocol of the DESFire EV1 is similar to that of the DESFire MF3ICD40 and was disclosed in [KvMOP11]. The power profile in Fig. 4.7a has been recorded during the second step of the protocol which involves two AES encryptions. The noticeable peaks in the signal arise from significant shifts in the DC component that might be caused by a countermeasure involving a switching of the internal power supply of the DUT [Plo09]. Another interesting feature is the increase in the signal amplitude from 500 µs to 1600 µs, which might result from the AES encryption.

German Electronic Passport

The German electronic passport [BSI] is based on a high-security contactless smartcard either manufactured by Infineon or NXP. It comprises several levels of security and is protected by different authentication mechanisms. We implemented the Basic Access Control (BAC) protocol [BSI10] which ensures that the device can only be read with the approval of the owner

by performing a 3DES-based mutual authentication based on a key derived from information printed inside the passport. SCA of the protocol itself would not provide a significant gain for an adversary (as the key is printed inside the document), however, it provides a starting point for the analysis of the DUT and allows to trigger a 3DES operation on the smartcard. As for the DESFire EV1, distinct offsets of the DC component can be observed, that again might stem from some protection mechanism, cf. Fig. 4.7b.

Mifare Ultralight C

This contactless smartcard was introduced by NXP in 2009 [NXP09] and targets cost-sensitive segments, e.g., paper tickets for public transport systems. Its cryptographic capabilities are limited, and the device only offers an authentication mechanism with 3DES (but no data encryption). Initially, we assumed that the DUT is based on a similar architecture as the Mifare DESFire MF3ICD40 analyzed in Chap. 7, however, this appears not to be the case: neither does the power profile resemble that of DESFire, nor are we currently able to reproduce the correlation results of Chap. 7.

Mifare Classic

Finally, we also examined the Mifare Classic, even though a successful key recovery by means of SCA would, in the light of the powerful cryptanalytical attacks [Cou09] that allow to extract the secret key in minutes, pose little additional threat. Thus, we only performed some superficial experiments, which, however, suggest that SCA could be utilized for practical key recovery as well. The trace depicted in Fig. 4.7d was acquired during the verification of the reader's response in the Mifare Classic authentication protocol [GKGM$^+$08]. The power profile exhibits eight characteristic peaks that appear to correspond to the eight bytes sent by the reader: in fact, we could observe a correlation with some bits of these values. Nevertheless, as the development of a complete attack would require substantial effort and at the same time is of rather limited relevance, we did not further investigate the susceptibility of Mifare Classic to SCA.

4.6 Conclusion

We presented analog demodulation circuits specifically designed for improving SCA of contactless smartcards. Our tools can be integrated into an existing EM SCA setup at a very low cost. The developed hardware allows to directly and instantly isolate the side-channel leakage from the reader signal before the digitizing step and hence significantly facilitates the alignment and further processing of SCA measurements of RFID devices.

Measuring the side-channel leakage of several real-world RFIDs, we verified the functionality of the developed circuits. In Chap. 7, the full-wave demodulator is applied for performing a successful side-channel attack on the popular Mifare DESFire MF3ICD40 contactless smartcard. In this context, the advantage of using an analog demodulator is further quantified by comparing the number of required traces for the key extraction to SCA solely based on digital demodulation performed in software

Chapter 5

GIAnT: A Platform for Implementation Attacks

Introducing our open-source fault injection platform GIAnT, we enable performing implementation attacks at a low cost. The FPGA-based GIAnT includes modules for controlling a wide variety of target devices (e.g., smartcards, µCs, or ASICs), for injecting faults by manipulating a DUT's supply voltage, and for recording side-channel traces, e.g., to perform an FI when a specific pattern has been detected in a trace. Illustrating the capabilities of the GIAnT, we analyze the vulnerability of two popular 8-bit µCs, the Atmel ATXMega256 comprising dedicated cryptographic hardware and an ATMega163-based smartcard, towards FI using voltage variations. We show that precisely timed modifications of the supply voltage allow to reliably skip instructions on the analyzed target devices.

Contents of this Chapter

5.1 Introduction

In this chapter, we describe the Generic Implementation Analysis Toolkit (GIAnT), a low-cost platform for performing non-invasive FI. The GIAnT is available to the community under the GNU General Public License (GPL) v2 to quickly put simple and more sophisticated FI attacks into practice. The open-source project [Sou13a] can be found at `http://www.sf.net/projects/giant`. The GIAnT is an extended and improved version of the FI framework presented in [KOP10], a joint publication with Timo Kasper.

5.2 System Overview

In this section, we introduce the building blocks of the FPGA-based GIAnT platform. We consider FI by means of manipulating the supply voltage. Hence, a suitable voltage source with precise control of the parameters for injecting the fault, i.e., the time instant of its occurrence, its width, and the voltage leading to erroneous behavior is required. One approach might be

to employ a commercial signal generator. However, these devices (with the required precision) cost several thousand USD and offer many features not needed in the context of fault attacks.

We decided to build a custom device using a DAC followed by an amplifier to directly generate the supply voltage with an arbitrary waveform. The use of similar designs is common practice in commercial evaluation labs [HCN+06, Bri10], however, details are generally not published and the devices are not sold (or very expensive [Bri10]). In contrast, our platform (including the schematics, software, and firmware) is available under an open-source license and can be built for approximately USD 300 (including all components and the PCB)[1]. The device is depicted in Fig. 5.1.

Figure 5.1: The GIAnT with an ATMega-based smartcard

The GIAnT is built around a Spartan6 FPGA, using a module manufactured by the open-source friendly start-up ZTEX [ZTE13]. The FPGA module provides 64 MB of fast Static Random Access Memory (SRAM) and a Cypress µC to realize the USB connection to a controlling PC. Both FPGA and µC can be programmed via USB with a framework provided by the vendor, allowing for high flexibility and quick reconfiguration.

5.2.1 Analog Input and Output

For the purpose of FI and recording (analog) side-channel signals, we developed a custom PCB, providing (amongst other components) an AD9708 DAC [Ana09c] for the manipulation of the supply voltage with a maximum update rate of 100 MHz. The width and time offset of the generated fault can thus be controlled with a precision of 10 ns, which is sufficient for our purposes, considering the relatively low clock frequency of most embedded systems. The overall structure of the system is depicted in Fig. 5.2.

The basic parameters of a FI by means of altering the supply voltage are depicted in Fig. 5.3. After the occurrence of the trigger signal, issued at a fixed time related to the execution of the targeted algorithm, the offset t_{offset} determines the exact time period passing before the fault

[1]The cost could be reduced as the supplier of the most expensive part (the FPGA module) offers discounts for open-source projects

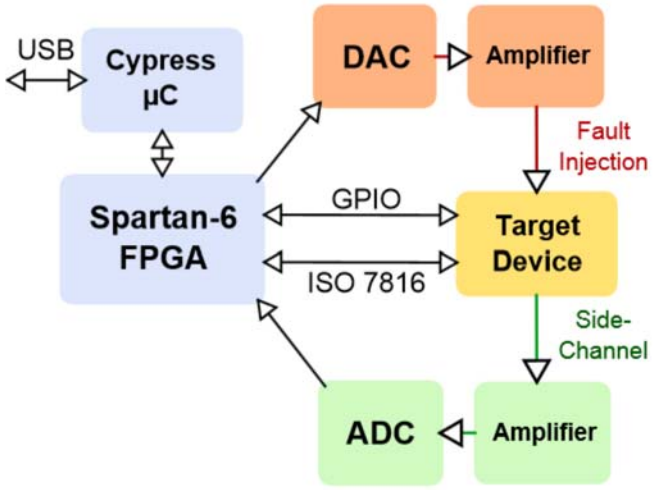


Figure 5.2: Basic structure of the GIAnT hardware

is injected. The duration of the fault is controlled by the parameter t_{width}. The voltage level during the fault is determined by V_{fault}. Depending on whether V_{fault} is higher or lower than the default supply voltage V_{default}, the fault is termed pulse or glitch, respectively.

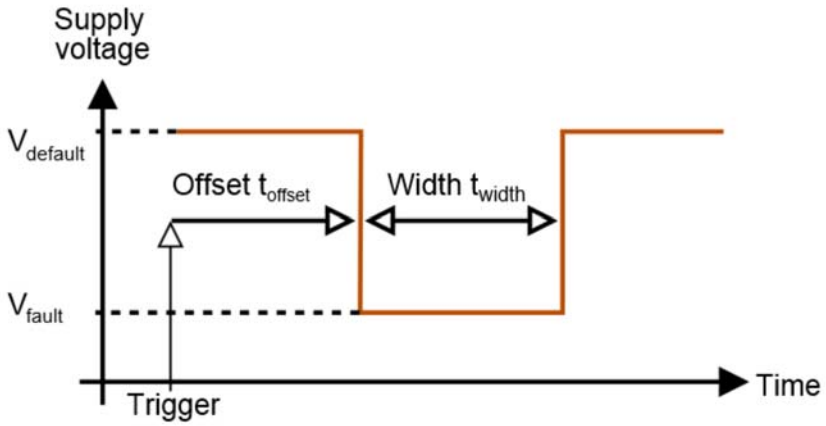


Figure 5.3: Parameters of a voltage glitch

Besides, the GIAnT incorporates an AD9283 ADC [Ana09b] for capturing analog signals with a maximum sample rate of 100 MHz. This allows for more complex attacks, e.g., dynamically triggering the injection of a fault in the case of asynchronous circuits [FML+03] by detecting characteristic patterns in the power consumption. We implemented a corresponding VHDL module continuously computing the sum of absolute differences between a previously set 128-point reference pattern and the recorded side-channel trace. When the computed difference is smaller than a configurable threshold, i.e., a similar pattern is detected in the trace, a trigger signal can be asserted.

We experimentally tested the pattern detection module with an Atmega163 (cf. Sect. 5.3.2) and were able to successfully locate patterns caused by EEPROM accesses on the DUT. Another interesting approach is the combination of FI with SCA to enable combined attacks as suggested in [AVFM07]. With the GIAnT, corresponding attacks can be experimentally verified with precise synchronization between the physical manipulation and the recording of the side-channel signal.

5.2.2 Communication Interfaces

The GIAnT includes a ISO 7816-compliant smartcard interface and the necessary firmware routines to communicate with such devices, either directly from the FPGA or from the controlling PC. The supply voltage of the smartcard can optionally be generated by the DAC for the purposes of FI. A variable resistor in the ground path allows power consumption measurements either using an external DSO or the built-in ADC.

In general, the design is kept modular to enable flexible inclusion of other FI and analysis methods, e.g., optical (laser) or EM techniques. Several GPIO pins are available to control additional hardware, e.g., a driver circuit connected to a laser diode. Besides, it is also possible to use these pins for interfacing to target devices, e.g., via typical serial links such as Serial Peripheral Interace (SPI) or Two Wire Interface (TWI).

To avoid the need for implementing a separate VHDL component for each new communication protocol, we designed re-configurable modules for serial input and output called `universal_rx_core` and `universal_tx_core`. These components operate at a frequency (configurable in steps of 762.94 Hz) between 762.94 Hz and 50 MHz that can be set via the PC API. For input, i.e., `universal_rx_core`, an external clock can be used alternatively, and the delay between the clock edge and the point when the data is sampled can be specified. The output module `universal_tx_core` provides a "data valid" signal (e.g., to enable external driver circuits) and can be used in an open-drain configuration, i.e., encoding a set bit with a high-impedance state. The data for both components is read from or written to 1024 byte First In First Out (FIFO) memories, which are accessible on the PC via the API.

In addition, the design contains two instances of the `universal_trigger_core`, a component to evaluate trigger conditions for an input pin. Amongst others, the module can detect rising and falling edges, level changes, or the signal being stable for a defined amount of time. Upon recognizing a trigger conditions, the component can generate a selectable event on the output pin (e.g., pull signal high or low, rising or falling edge, switch signal level) with a user-defined output delay of up to 85.9 s (in steps of 20 ns).

The combination of `universal_rx_core`, `universal_tx_core`, and `universal_trigger_core` was used to establish the communication to several DUTs in this thesis, including the SimonsVoss door lock (Chap. 8) and the Maxim DS2432 and DS28E01 ICs (Chap. 10). Note that implementing the respective protocols did not require modifications of the VHDL code and hence no re-synthesis of the FPGA design. All necessary configuration steps were performed by the controlling PC.

Finally, we also implemented an ISO 14443 A [Int01b] reader for 13.56 MHz RFIDs, cf. Sect. 6.3.1. To this end, we directly generated the necessary 13.56 MHz waveform using a sine generator (implemented with an IP core provided by Xilinx), converted it to an analog signal using the DAC, and output it using a PCB antenna originally used for the custom RFID

reader described in Sect. 3.1.2. The modulation of the reader waveform (i.e., switching the signal on and off) was performed using the (digital) output of the `universal_tx_core`. For the backward channel from the transponder to the reader, we used the full-wave rectifier of Sect. 4.3 and digitized the demodulated signal using the ADC of the GIAnT. The further demodulation (i.e., distinguishing one and zero bits) was then performed in software on the controlling PC. We verified the basic functionality of the RFID module by practically implementing and perfoming the authentication protocol with a Mifare DESFire MF3ICD40 card, cf. Chap. 7.

5.2.3 Programming and Control Interface

The GIAnT appears to the PC as a set of 32 read-only and 64 write-only 8-bit registers. More precisely, the PC communicates with the Cypress µC via USB, and the µC in turn writes the data to the Spartan 6 FPGA. However, the read and write operations to the registers of the FPGA are—from the point of view of the API on the PC—transparently executed. The basic functionality is wrapped in the C++ class `fault_fpga_spartan6` that, amonst others, provides methods to initialize and perfom the USB communication using the `libusb` library and to access the FPGA's registers. In addition to the register-oriented operations, a high-speed read mode for the external 64 MB SRAM is available, providing a data rate of approximately 20 MB/s, e.g., to acquire measurements recorded with the ADC.

Additional C++ classes implement the low-level register accesses required for controlling the modules of the GIAnT. For example, the DAC is configured using the class `dac` and the smartcard interface is encapsulated in `smartcard`. For a detailed documentation, the GIAnT project includes a `doxygen` documentation [Sou13a]. To illustrate the basic usage, the code in Listing 5.1 exemplarily demonstrates the communication with a smartcard and the injection of a voltage fault.

```
1  // Control objects
2  dac vfi;
3  smartcard sc;
4
5  // Computes voltages for +-8 V supply
6  const double v_step = 16.0/256.0;
7  const uint8_t v_5 = static_cast<uint8_t>(-5.0/v_step + 128.0);
8  const uint8_t v_neg_5 =  static_cast<uint8_t>(5.0/v_step + 128.0);
9
10 // Set DAC voltages
11 vfi.setLowVoltage(v_5);
12 vfi.setHighVoltage(v_neg_5);
13 vfi.setOffVoltage(128);
14
15 // Setup trigger after last bit sent
16 vfi.setTriggerEnableState(
17         fault_fpga_spartan6::FI_TRIGGER_CONTROL_SC_SENT, true
18 );
19
20 // Prepare APDU
```

```
21 byte_buffer_t data;
22 data.resize(16, 0);
23 byte_buffer_t test_apdu =
24         smartcard::makeT1Apdu(0x80, 0x40, 0x0, 0x0, data, 16);
25
26 // Init smartcard
27 byte_buffer_t atr = sc.resetAndGetAtr();
28
29 // Prepare fault injection
30 vfi.clearPulses();
31
32 // Add pulse, 3000 ns after trigger, width 100 ns
33 vfi.addPulse(3000, 100);
34
35 // Arm fault injection
36 vfi.arm();
37
38 dbg::out(dbg::info) << "TX: ";
39 util::hexdump(dbg::out(dbg::info), test_apdu);
40 dbg::out(dbg::info) << std::endl;
41
42 // Send and receive data
43 byte_buffer_t rx = sc.handleRxTx(test_apdu);
44
45 dbg::out(dbg::info) << "RX: ";
46 util::hexdump(dbg::out(dbg::info), rx);
47 dbg::out(dbg::info) << std::endl;
```

Listing 5.1: Example C++ code for smartcard communication and FI

On line 2, the C++ objects for controlling the relevant modules of the GIAnT are initialized. Starting with line 11, the voltages for the voltage pulse are configured, whereas V_{default} is set to 5 V (`setLowVoltage()`) and V_{fault} to -5 V (`setHighVoltage()`). To trigger the FI, on line 16 an internal signal that is asserted after the last bit of a command has been sent to the smartcard is activated. Then, an Application Protocol Data Unit (APDU) for the test smartcard is prepared on line 21, invoking an encryption operation in this case. Before sending this command, the smartcard is initialized on line 27, reading the Answer To Reset (ATR). The actual parameters for the timing of the FI are specified starting on line 30, setting up a single pulse with 100 ns width and 3 µs offset to the trigger. After arming the FI module on line 36, the next communication leads to the injection of the specified fault. The call to `handleRxTx()` on line 43 sends the prepared APDU to the DUT, thus triggers the FI, and receives the response of the smartcard. This result of the communication on the DUT can then be further evaluated, e.g., to determine whether a fault was successfully injected.

5.3 Practical Verification

The setup used for practically verifying the functionality of the GIAnT is shown in Fig. 5.4. In this case, the controlling PC is connected to a DUT via a serial interface to start a computation on the target and receive the result. Note that the communication modules of the GIAnT, cf. Sect. 5.2.2, could have been utilized as well. The USB link is first used to program the FPGA, after that, a suitable PC applications communicates with the GIAnT and, e.g., sets the parameters for the FI.

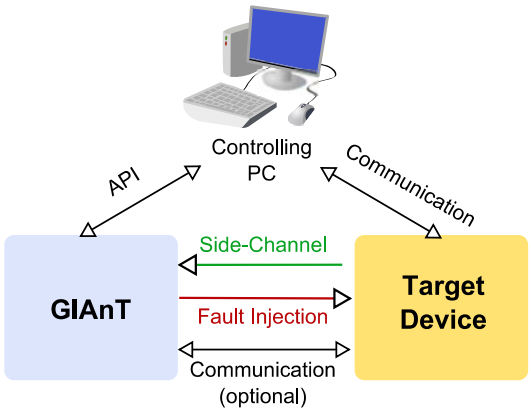


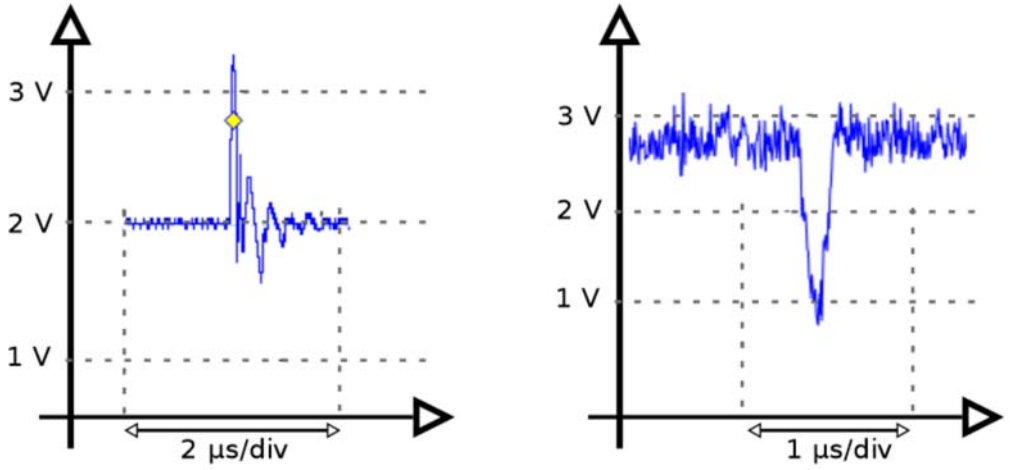
Figure 5.4: Hardware setup for the practical FI on two Atmel µCs

In this section, we pinpoint the precise character of a successful FI for two simple target devices, an Atmel ATXMega256 (cf. Sect. 5.3.1) and an Atmega163 (cf. Sect. 5.3.2). We give practical results for attacking the 3DES hardware of the Atmel ATXMega256 and software implementations on both DUTs using voltage faults. The main goal of this section is to demonstrate that the GIAnT can be used for practical attacks. The susceptibility of the Atmel devices to implementation attacks like SCA and FI is well known and hence, we used these devices for testing our framework.

For determining the right choice of pulse or glitch, many permutations of the parameters for a voltage fault, cf. Fig. 5.3, need to be practically tested while surveying the behavior of the target device and determining whether a fault was successfully injected or not. Multiple faults, i.e., several subsequent pulses would allow for even more powerful attacks and can also be generated with our setup. Experimentally, we found suitable parameters for a successful FI for both target devices. The according waveforms (measured at the supply pin of the DUT) are depicted in Fig. 5.5. In the following Sect. 5.3.1 and Sect. 5.3.2, we further detail on the results of the experiments for the ATXMega256 and the ATMega163, respectively.

5.3.1 Target Device 1: ATXMega256

The ATXMega256 (Revision A3) is an 8-bit µC developed by Atmel [Atm]. It provides 256 kB flash memory, 16 kB SRAM, and a clock frequency of up to 32 MHz. Furthermore, 32 8-bit registers are available. A special feature of this µC is the cryptographic coprocessor which can be used to perform (3)DES and AES en- and decryption with hardware acceleration. Initial tests regarding the susceptibility of the device to FI were carried out with simple programs



(a) ATXMega256 (b) ATMega163

Figure 5.5: Voltage waveform used for FI on the DUTs

consisting of loops executing a few CPU instructions. To provide results for cryptographic algorithms, we also implemented both CRT-RSA (in software) and 3DES (using the internal cryptographic hardware engine).

The DUT was placed on a testing PCB designed for the purposes of FI and SCA, i.e., the board only features simple interface logic, contains no buffer capacitors, and offers direct access to the power pins of the device. The µC ran at the default clock frequency of 2 MHz, generated by the internal oscillator. This corresponds to a time of 500 ns per instruction, since the processor executes one instruction per clock cycle.

There are three possible outcomes of attacking a DUT with voltage faults: in the first case, an implemented algorithm returns the expected correct result, implying that the fault did not affect the device, e.g., because the fault was induced at a wrong position. In case two, the device returns zero, e.g., because the fault occurred during an input/output operation or because the device was completely reset. Case three is a useful fault, i.e., a fault that changes the expected behavior of the device in a way that allows to for instance break a cryptographic algorithm.

In our initial experiments, it turned out that the ATXMega is susceptible to a short increase of the supply voltage from $V_{\text{default}} = 2\,\text{V}$ to $V_{\text{fault}} = 3.3\,\text{V}$ for a pulse width of 10 ns. The actual waveform at the power pin of the µC is shown in Fig. 5.5a. Further investigation with respect to the effect of a fault showed that the FI causes the CPU to skip an instruction. This result has been reported before in the literature for other µCs, cf. [HCN+06, KOP10], and allows to attack cryptographic algorithms as well as other protection mechanisms, e.g., password checks[2].

For an implementation of CRT-RSA, we could carry out both the Bellcore [BDL97] and the Lenstra [BDH+97], attack by, for example, skipping a single instruction within a long number multiplication, and hence successfully recover the private key of a signature computation.

Note that the device is very sensitive to the amount of change of the supply voltage: in our experiments, there was only a rather small set of pulse parameters that allowed us to inject the

[2]By skipping the respective comparison operations

desired fault. Moreover, glitches (i.e., reductions of the supply voltage) either caused no effect or resulted in a restart of the µC. The device reliably detects undervoltage conditions, even if they occur only for a fraction of a clock cycle. Interestingly, turning off the built-in brown-out detection of the µC did not change this behavior.

We carried out initial experiments involving multiple faults indicating that the µC is susceptible to this approach as well. Note that these faults are subject to certain restrictions, e.g., that it is not possible to skip two consecutive instructions: if the duration between the pulses is too short (i.e., within a few clock cycles), the device is reset regardless of the choice of parameters. However, it appears to be feasible to skip instructions that have a certain minimum time distance.

Performing our experiments at a clock frequency of 2 MHz does not pose a limitation for real-world attacks. Due to the design of the clock system of the ATXMega, the device always initially operates at a default frequency of 2 MHz. A different clock source is selected in software, i.e., by setting certain registers in the initialization routines of a program. Hence, the respective instruction forcing the µC to switch to a different frequency can be skipped by injecting faults during the initialization, in order to later inject a fault during the computation of the cryptographic algorithm as detailed above.

Instruction-Skip Fault Attack on DES

As mentioned, the ATXMega contains a hardware engine for accelerating DES operations. For this purpose, the device provides the instruction `DES k`, where `k` is a counter for the current round of the algorithm starting at zero. Before executing this instruction, the plaintext and the key are stored in certain CPU registers. Then, `DES 0`, `DES 1`,..., `DES 15` are executed and the resulting ciphertext is stored in the registers that previously held the plaintext.

As for other CPU instructions, the specified voltage pulse was employed to skip `DES k`, i.e., omit one round of the full DES. To exploit this effect, an analysis similar to that in [BS97, MRL+06] was employed. We adapted this approach to first completely recover the round key of the sixteenth DES round using on average only three different ciphertext/plaintext pairs (and the according faulty ones). The attack was then extended to make use of the results when skipping previous rounds to obtain the complete key of a full DES. Finally, with the extended approach, we were also able to uniquely recover the whole key used for a 3DES encryption.

Let c denote the DES encryption of a message x with key K, and c' denote an encryption of the same message where the final round has been skipped. Since the inverse of IP at the end of the computation can easily be inverted, the attacker knows L_{16} and R_{16} by applying IP to the correct ciphertext c. Because the final round is omitted during the computation of the faulty ciphertext c', the adversary is in possession of L_{15} and R_{15} as well. The aim is to reconstruct the 48-bit round key K_{16} of the final round, knowing c and c'—the problem essentially reduces to attacking a single-round DES.

The standard DES round function f, illustrated in Fig. 5.6, takes K_{16} (with a length of 48 bit) and R_{15} (with a length of 32 bit) as input. R_{15} is expanded to 48 bit by the permutation E and XORed with K_{16}. The result is input to eight non-linear S-boxes which provide a 32-bit output, followed by another permutation P. Note that with R_{15} and with $L_{15} \oplus L_{16}$, both the output of the round function f and one of the inputs are known.

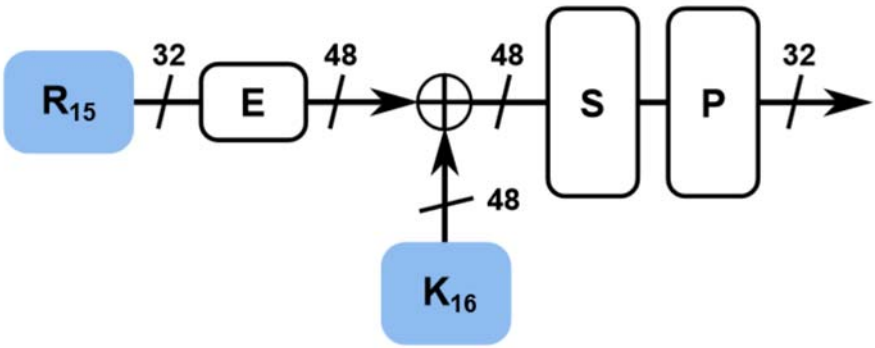


Figure 5.6: Round function of the DES

The only unknown value is the second input K_{16}. To compute it, the permutation P is inverted to obtain the output of the S-boxes, and R_{15} is expanded by applying E. Note that the S-boxes can not be directly inverted since they are not injective for DES. To obtain the input, each S-box needs to be handled separately: if the output of the S-box is known, all of its possible inputs can be computed with an uncertainty of only two bits. Thus, the attacker obtains four possible input candidates for each S-box. In total, there are $4^8 = 2^{16}$ candidates to be considered for all S-boxes. Every candidate is now XORed with the extended R_{15} to obtain all possibilities for K_{16}, implying a brute-force complexity of 2^{16} DES_K operations.

Now, we can use these 2^{16} candidates for $K_{16} = PC2\,(C_{16}, D_{16})$, where C_i and D_i denote the two 28-bit halves of the key schedule state, to compute the actual 56-bit DES key. In order to determine the latter, the key schedule of DES has to be inverted, starting with PC2. PC2 is a permutation from 56 bit to 48 bit, thus, guessing the input results in another eight bits of uncertainty. As a result, 2^{24} candidates for the 56-bit key remain for the exhaustive key-search..

To improve this approach, we assume that the attacker is in possession of N correct ciphertexts $c_1, c_2, \ldots c_N$ and the according faulty ones $c'_1, c'_2, \ldots c'_N$, calculated with different unknown plaintexts $x_1, x_2, \ldots x_N$ but the same key. Hence, the step for obtaining candidates for K_{16} can be repeated using different pairs of c_n and c'_n. The attacker can compute N sets $A_1, A_2, \ldots, A_N$ ($|A_i| = 4\ \forall\ i$) of input candidates for each S-box. If N is big enough, the intersection of the sets $A = A_1 \cap A_2 \cap \cdots \cap A_N$ contains only one element. In this case, the round key K_{16} can be determined exactly without using brute force. Thus, the brute force complexity to obtain the secret key decreases from 2^{24} to 2^8.

Moreover, knowing the full subkey K_{16}, the final round can be inverted, i.e., given any (L_{16}, R_{16}) one can also compute (L_{15}, R_{15}). Now, assume that not the final (16th) but the 15th round is skipped. Hence, the input to round 16 is (L_{14}, R_{14}), instead of (L_{15}, R_{15}), and accordingly L_{14}, R_{14} can be obtained by removing round 15 knowing K_{16} and the (faulty) output. Similar to the analysis of the final round, it is then possible to recover the round key K_{15}. Iterating this approach, all round keys can be recovered until the full key K of the DES operation is uniquely determined, without the need for brute-force. This is advantageous in the case of 3DES: one first determines the key K^1 for the third DES encryption, then continues with the same attack for the middle decryption—this time skipping rounds of this operation—and then handles the first encryption to finally obtain the full 168-bit key.

Table 5.1 shows example results yielded by the successful injection of a fault. The DES key was set to `1f f1 1f 3e 00 ef 33 22`, the plaintext to `00 00 00 00 00 00 00 00`, resulting in an (expected) valid ciphertext `9f eb 63 78 6a 3a dc 85`. A simulation confirmed that the erroneous values of Tab. 5.1 are indeed computed when skipping a specific round. An interesting fact resulting from the obtained outputs is that some rounds yield the same (faulty) ciphertext when they are skipped. This suggests that the implementation of the DES hardware engine both realizes the execution of the round function and the key schedule update. Since the amount of rotations in the key schedule is the same for many rounds, the skipping of subsequent rounds (with the same amount of rotations) results in the same output. However, this does not cause a problem with the extended approach, as all 56 (non-parity) bits of the key are used in either round fifteen or sixteen.

Offset	Width	Ciphertext	Round
2625	90	`339e8a6d473cbce9`	1 - 2
3125	90	`7e4cffe3f37ba231`	3 - 8
6125	90	`fb8955c63025d5d3`	9
6625	90	`cff66e5cda1a81d2`	10 - 15
9625	90	`cffb372c6f2fc8c1`	16

Table 5.1: Example results of the fault attack on DES

Another characteristic of the implementation of the DES engine is that the value `cf fb 37 2c 6f 2f c8 c1` is not equal to the 15th round state following the standard DES: the IP has to be applied to this value to obtain the correct round state. As stated in the manual [Atm], the `DES k` operation performs IP before and the inverse after each round. This is a common optimization to avoid the need to separately handle IP.

5.3.2 Target Device 2: ATMega163 Smartcard

The second target ATMega163 is an instantiation of Atmel's ATMega architecture [Atm03], embedded in a standard ISO 7816-compliant smartcard (this product is known as "Open Platform" or "Funcard", especially in the pay TV sector, where it was often used for emulating original cards). Our FI platform is designed to support smartcards, hence we inserted the card without any modifications into the slot provided (cf. Fig. 5.1) to perform the FI.

In contrast to the ATXMega, the computational and storage capabilities of the ATMega are more restricted (16 kB Flash, 1 kB SRAM, max. 8 MHz clock frequency) and the device does not feature special accelerators for cryptographic algorithms. Thus, apart from the simple test programs, we only implemented CRT-RSA on this platform. Similar to the ATXMega, the controller was operated at a clock rate of 2 MHz supplied by the smartcard reader module.

For the ATMega, similar results as for the ATXMega were obtained, i.e., with suitable settings, CPU instructions could be reliably skipped and a CRT-RSA implementation was successfully broken. However, compared to the ATXMega, the parameters of the fault waveform differed: Instead of a voltage pulse, a glitch ($V_{\text{default}} = 2.8$ V, $V_{\text{fault}} = 0.89$ V, $t_{\text{width}} = 100$–200 ns, cf. Fig. 5.5b) turned out to cause the desired skipping of an instruction. Again, the correct parameters were found by exhaustively searching a range of values using the GIAnT.

5.4 Conclusion

With the GIAnT, we introduced a low-cost tool for active and passive implementation attacks. The GIAnT can, amongst others, communicate with a variety of DUTs (including smartcards, ICs with serial interfaces, and RFIDs), inject faults via manipulations of the supply voltage, and record side-channel traces using the built-in ADC.

Validating the functionality of the GIAnT, we show that both the ATMega163 and the ATXMega256 µC are highly susceptible to FI attacks. Our findings enable an adversary to break cryptographic algorithms implemented in software (CRT-RSA) and in hardware (using the DES engine of the ATXMega). On the basis of the demonstrated skipping of instructions by manipulating the supply voltage, other attacks against non-cryptographic (but security-relevant) mechanisms can be realized, e.g., by removing a comparison in a password check. The FI attacks can, once the parameters have been determined, carried out within seconds, and even the injection of multiple faults, i.e., subsequent faults during one execution of an algorithm, is possible.

Using the GIAnT, implementation attacks can in many cases be put into practice at a very low-cost of approximately USD 300, without requiring expensive lab equipment. Since the device is released under an open-source license, interested developers can contribute additional functionality or utilize the existing framework for testing the susceptibility of cryptographic implementations towards SCA and FI attacks.

Part III

Real-World Attacks

Chapter 6

RFID Range Measurements

This chapter deals with security threats concerning RFID and other wireless devices in general. We introduce a setup for monitoring the communication of active and passive RFIDs for actively communicating with the devices at a range exceeding the original specification. We investigate an active RFID transponder operating at 433.92 MHz and three passive 13.56 MHz RFIDs in order to determine how far the specified coverage can be extended. Assuming an adversary with limited know-how and funds, we conduct all analyses solely with commercially available low-cost and easy-to-use equipment. For the passive RFIDs, we reproduce the results of previous studies and show that the reading ranges can be increased by a factor of three to five, depending on the device's technology. Furthermore, we demonstrate that the ranges for the active RFID transponder can be increased by up to a factor of eight. Hence, this type of RFID is particularly vulnerable to attacks from a distance from between 500 m and 1 km.

Contents of this Chapter

6.1 Introduction

Passive RFID devices are commonly employed for access control, ticketing, and identification purposes, e.g., in the form of contactless smartcards operating at a frequency of 13.56 MHz (HF) or car immobilizers operating at 125 kHz (Low Frequency (LF)). The devices possess no own energy supply, e.g., a battery, and are solely powered by the EM field of an interrogating reader device, which implies, amongst others, a limited operating range in the order of a few meters.

In contrast, *active* RFIDs are usually battery-powered and achieve operating ranges in the order of tens or hundreds of meters. They often communicate at ISM frequencies in the UHF range, i.e., in Europe very often at a frequency around 433 MHz. According devices serve for instance as remote controls for RKE systems, e.g., for opening cars (often combined with a passive transponder in the car key). Actively-powered solutions are required in all situations where

a high reading range or a short interrogation period is required and hence passive transponders are no option, e.g., license plates of vehicles that are queried while driving past at a high speed. Furthermore, active RFIDs are often equipped with sensors for monitoring and recording various environmental conditions, such as temperature or humidity, and provide security features, such as the capability to trigger a real-time alarm upon the detection of an intrusion by means of light or shock sensors. These advanced transponders are mainly used in the supply chain for tagging containers with large assets, e.g., for the transport of medical supplies, food, and other goods in order to track their route while assuring that certain environmental conditions are met.

In this chapter, we investigate both passive HF and active UHF RFIDs and analyze how much the (specified) communication ranges can be extended both for a passive eavesdropping attacker and an adversary actively querying the DUTs. For estimating the real-world threat of our findings, we assume an attacker with limited skills and funds, e.g., an electronic hobbyist or a criminal with electrical engineering background. Accordingly, for all practical analyses presented in this chapter, we employed no special equipment or know-how but consistently opted for low-cost and easy-to-use solutions based on equipment that is commercially available. Consequently, we suppose that our results for the eavesdropping and detection ranges can be further improved by a well-funded and highly skilled adversary (or organization).

The analysis of the active UHF transponders was performed together with Timo Kasper and published at SoftCom 2011 [KOP11a].

6.1.1 Related Work

The topic of ranges for eavesdropping on RFID communication has been discussed in the literature for the case of passive RFIDs. Especially in the case of identification documents based on contactless smartcards, such as the electronic passport (ePass), eavesdropping poses a severe threat for the privacy of individuals [LKLRP07] and has been thoroughly analyzed: the active operating range (specified with 8 to 15 cm) can be increased up to approximately 30 cm [FK, KW06], while passively monitoring the communication in both directions is practically feasible from a distance of several meters [Han06, Han08]. Simulations by NXP mention the possibility of eavesdropping from a maximum of 50 m [NXP07].

In contrast, to our knowledge, the actual ranges for eavesdropping on active RFIDs have never been practically evaluated in the literature, despite the fact that a common interest in the topic is evident: for instance, the attack [EKM+08] on the widespread RKE system KeeLoq allows to extract the secret keys of remote controls (and consequently produce duplicates, open doors, etc.) by eavesdropping on one single (encrypted) transmission. The results of practical eavesdropping on active RFIDs are comparable to KeeLoq remote controls that are operating at the same frequency, and hence allow to assess the real security risk resulting for such active wireless devices. In general, evaluating the vulnerability of active RFIDs is of great importance, since the devices are employed in various security, privacy, and safety-relevant applications, in which the system integrators are often not aware about the threats discussed in the following.

6.1.2 Contribution

We briefly introduce security threats and attacks on wireless devices in Sect. 6.2, pinpoint the relevance of eavesdropping and then evaluate the susceptibility of both passive 13.56 MHz and

active 433.92 MHz RFIDs with respect to this threat. For this purpose, we introduce fundamentals about passive and active RFIDs technology in Sect. 6.3.1 and Sect. 6.4.1, respectively. We detail on low-cost setups to passively monitor the communication and actively query the DUTs in Sect. 6.3.3 (for 13.56 MHz) and Sect. 6.4.3 (for 433.92 MHz). In Sect. 6.4.4 and Sect. 6.3.4, we conduct practical experiments that result in determining a lower bound of the range for eavesdropping and active communication. We conclude in Sect. 6.5, evaluating the threat posed by the demonstrated methods in practice.

6.2 Security Threats for RFID

The wireless communication channel used by embedded devices such as RFIDs is subject to a plethora of attacks. The *detection and tracking* of persons or objects equipped with an RFID endangers the privacy [LKLRP07] and may furthermore enable targeted assaults: in [JMW05], the notion of an "RFID bomb" was coined, which is triggered only if a (specific) transponder is present.

More elaborate methods in this regard include *passive eavesdropping*, i.e., the attacker monitors the communication between the RFID and an authorized reader, and *active reading*, i.e., the adversary communicates with and reads the information stored on an RFID transponder clandestinely.

The exchanged data may then be used in a *replay attack* in which the recorded communication is re-sent to simulate the presence of the genuine transponder. Replay attacks can be prevented by employing cryptographic schemes, e.g., a challenge-response protocol, that establish mutual authentication between reader and transponder.

However, *MITM* or *relay attacks* allow to overcome systems using cryptographically secure authentication. Relay attacks on RF devices were first described in [KW05] and put into practice for RFIDs in [Han05, KCP07]. To carry out a relay attack, the adversary uses two devices connected with a high-range wireless link. One device simulates a transponder to the genuine reader, e.g., a payment terminal or an access control reader. All requests sent by the genuine reader are forwarded over the wireless link to the other device that is placed close to the victim. This device acts as a reader for, e.g., an RFID in the victim's pocket, transmits all queries to this RFID, receives the (valid) responses, and sends them back via the wireless link. The device on the other end of the relay is then able to appear as the genuine transponder of the victim and, e.g., carry out a payment or open a door lock.

For the general threat of a *Denial-of-Service (DoS) attack* that applies to any interconnected system, several special cases exist in the context of RFIDs. *Sleep deprivation* exhausts the battery of an active transponder by continuously sending requests, a scenario that is usually not taken into account when a manufacturer specifies the battery life. Certain RFIDs can be permanently deactivated with a so-called "kill" password. If this password is identical for many transponders and can be obtained by an adversary (cf. for instance [OS07]), this enables an attack in which an entire system could be disabled by sending a single command, possibly from a large distance. Finally, RFIDs may also serve as a carrier for *malware* [RCT06]. If certain data on a transponder can trigger a malfunction of a reader, e.g., due to a buffer overflow, an adversary may be able to gain control over the system. The "infected" reader could then copy the malicious code to every (genuine) transponder it communicates with, which in turn would infect other readers.

For practically conducting the described attacks in the real world, the achievable ranges for actively communicating with the transponders and for passively eavesdropping on the communication play an important role. The larger the distance from which the attacks can be carried out, the more severe is the resulting threat. For example, RFIDs in a warehouse can be considered secure if their maximal operating range is in the order of meters, but become subject to attacks if the range can be increased to for example 100 m, i.e., communication is possible from outside the warehouse.

6.3 Passive HF RFID at 13.56 MHz

In this section, we deal with methods to increase the passive eavesdropping and active communication range for passive RFIDs operating at 13.56 Mhz. Although this issue has been investigated before [Han08, Han06], we attempted to reproduce the results of previous work and especially aimed at analyzing the ranges for a modern, feature-rich transponder, the Mifare DESFire EV1.

6.3.1 Fundamentals

All passive RFIDs considered in this section operate at a frequency of $f_c = 13.56\,\text{Mhz}$. The wireless interface is implemented according to either ISO 14443 [Int01b] or ISO 15693 [Int09]. In both cases, the transponder is wirelessly powered via inductive coupling between the readers's antenna and the counterpart in the transponder. Besides, the same wireless link is used for communication between reader and transponder. The main difference between the two standards is the (specified) operation range. While ISO 14443-based systems specify an operation range of less than 10 cm, certain ISO 15693 transponders can work for a distance of up to 1.5 m between transponder and readers. Thus, ISO 14443 is sometimes refered to as "proximity coupling", whereas ISO 15693 is called "vincinity coupling".

Technical Details of ISO 14443

ISO 14443 defines two sub-standards for the employed modulation schemes and the data coding [Int01a]: For ISO 14443 A, the data sent by the reader (R $\rightarrow$ T) is transmitted using 100% Amplitude-Shift Keying (ASK) (i.e., On-Off-Keying (OOK)) and encoded using a modified Miller code with a data rate of $106\,\text{kBit/s}$. The backward channel from the transponder to the reader (T $\rightarrow$ R) uses a 10% load modulation of the 13.56 MHz carrier to generate a subcarrier at $\pm^{13.56\,\text{MHz}}/_{16}$. This subcarrier is modulated with the Manchester-encoded data at a rate of $106\,\text{kBit/s}$. ISO 14443 B employs a 10% ASK to transmit Non-Return-to-Zero (NRZ) encoded data from the reader to the transponder at $106\,\text{kBit/s}$. The transponder's data is encoded using NRZ-L and sent using load modulation, resulting in a Binary Phase Shift Keying (BPSK) modulated subcarrier at $\pm^{13.56\,\text{MHz}}/_{16}$.

Both standards include additional mechanisms to detect transmission errors, e.g., parity bits and in many cases Cyclic Redundancy Check (CRC) or XOR checksums. Apart from that, both ISO 14443 A and B include mechanisms for anticollision and other functionality on higher protocol layers [Int01b, Int01c]. Additionally, the standards also allow for higher data rates up to $848\,\text{kBit/s}$ after the initialization and anticollision protocol has been executed.

In the following, we deal with transponders compliant to the today far more prevalent ISO 14443 A. In general, since the basic communication principle (inductive coupling) is the same in both substandards, it is likely that the results we obtained for ISO 14443 A also hold for B-type RFIDs.

Technical Details of ISO 15693

ISO 15693 [Int09] (also known as ISO 18000-3) employs an either 10% or 100% ASK for the channel R → T. Data is encoded in the position of a pulse, whereas either a one-of-four or a one-of-256 coding is possible. The data rate is 26.48 kBit/s and 1.65 kBit/s, respectively. The transponder sends data by load-modulating the field generated by the reader using an ASK with either one subcarrier ($^{13.56\,\mathrm{MHz}}/_{32}$) or two subcarriers ($^{13.56\,\mathrm{MHz}}/_{28}$ and $^{13.56\,\mathrm{MHz}}/_{32}$). For encoding the data, a Manchester code is employed. Two modes "low" and "high" are available for the data rate, resulting (together with the choice of the subcarrier) in the data rates given in Table 6.1.

Mode	One subcarrier	Two subcarriers
Low	6.62 kBit/s	6.67 kBit/s
High	26.48 kBit/s	26.69 kBit/s

Table 6.1: Data rates for ISO 15693 transponders

Devices Under Test

For our experiments, we used three DUTs that are representative for different classes of passive RFIDs in our opinion. DUT 1 is an ISO 14443-compliant transponder developed approximately around 2000 with a proprietary encryption mechanism[1]. As an example for a modern RFID smartcard, we selected the Mifare DESFire EV1 [NXP10b] following ISO 14443 as DUT 2. Both devices have a specified read range of 10 cm. DUT 3 is an ISO 15693-compliant device[1] with a specified operation range of up to 60 cm, also using a vendor-specific cryptographic algorithm.

6.3.2 Theoretical Background

For HF RFID systems at $f_c = 13.56\,\mathrm{Mhz}$, transponder and reader form an inductively-coupled system operating in the near field. For the typically relatively small distances d between reader and transponder, the wavelength condition $^{\lambda}/_{2\pi} < d$ (with wavelength $\lambda = {}^{c}/_{f_c} = 22.11\,m$) holds [Fin03]. Thus, for distances less than $^{\lambda}/_{2\pi} = 3.52\,m$, the system essentially behaves like a transformer. A change at one antenna instantly affects the other antenna, as no freely propagating EM wave has formed yet. There are several analytical expressions given in [Fin03] that allow to estimate the effect of the distance between transponder and reader in simplified scenarios. First, the strength of the H field along the middle axis of a round loop antenna is given as

[1] Due to confidentiality reasons, we cannot disclose the exact product name and the vendor

$$H(d) = \frac{i_R \cdot N_{ant} \cdot r_{ant}^2}{2\sqrt{\left(d^2 + r_{ant}^2\right)^3}} \tag{6.1}$$

for $d < \lambda/2\pi$ with d the distance to the center along a line orthogonal to the antenna's plane, i_R the current flowing through the antenna, N_{ant} the number of windings, and r_{ant} the antenna's radius. The field strength falls $\propto \frac{1}{d^3}$, i.e., by 60 dB per decade. For a square loop, a similar expression holds:

$$H(d) = \frac{i_R \cdot N_{ant} \cdot a_{ant}^2}{2\pi\sqrt{\left(\left(a_{ant}/\sqrt{2}\right)^2 + d^2\right)\left(\left(a_{ant}/2\right)^2 + d^2\right)}} \tag{6.2}$$

with a_{ant} the length of one side and the other parameters defined as for Eqn. 6.1. The H field determines the maximum operating range: For a given transponder, a minimum field strength H_{min} at which the device begins to work correctly can be computed, cf. [Fin03, p.81]. Obviously, there are several ways to increase the field strength and thus the operating range: The number of turns N_{ant} and the current i_R affect H linearly. The antenna radius r_{ant} (or the length of one side a_{ant}) enters both in the numerator and the denominator. This leads to the effect that—for a given antenna with N_{ant} and i_R—the field strength H at a given distance d is maximized for an antenna radius of $r_{opt} = \sqrt{2} \cdot d$.

The inductively-coupled system of the reader and the transponder can be modeled as the circuit depicted in Fig. 6.1. Here, L_R and L_T are the inductances of the reader and transponder antenna, respectively, R_T the resistance of the transponder's antenna, C_T the accumulated capacitance of the device, and R_L the load resistance representing the transponder's circuitry. M is the mutual inductance between the antennas, which is, for the simple case of wire loops, like the H field proportional to $\frac{1}{d^3}$.

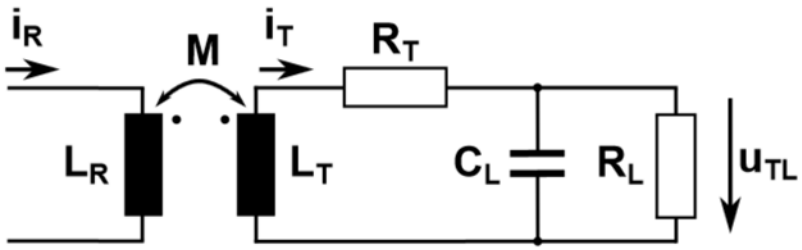


Figure 6.1: Equivalent circuit of the inductive coupling between reader and transponder [Fin03]

The voltage u_{TL} at the input of the transponder's circuitry can then be written as a function of the current i_R in the reader antenna:

$$u_{TL} = \frac{j\omega M i_R}{1 + (j\omega L_T + R_T)\left(1/R_L + j\omega C_T\right)} \tag{6.3}$$

To identify the influence of an eavesdropping antenna, the circuit of Fig. 6.1 can be modified to include a third coupled inductance L_P that is the probing antenna. For simplicity, we merged the components of the transponder and the eavesdropping antenna in one impedance Z_T and

Z_P, respectively. Now, three mutual inductances M_{RT}, M_{RP}, and M_{PT} have to be considered, since the probe antenna may also interact with the other components. The respective circuit is shown in Fig. 6.2.

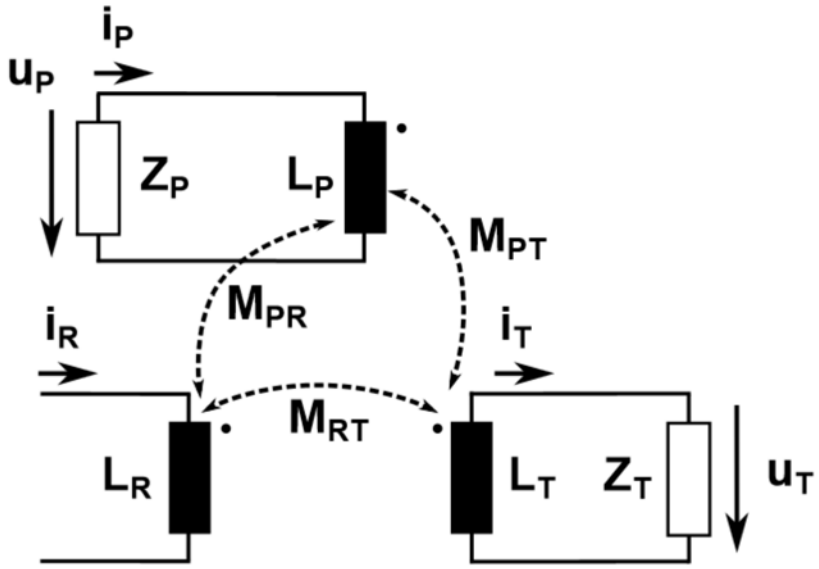


Figure 6.2: Equivalent circuit of the inductive coupling between reader, transponder, and an eavesdropping antenna

From Faraday's law of induction, it follows that

$$\begin{aligned} u_T &= M_{RT}\frac{di_R}{dt} - L_T\frac{di_T}{dt} + M_{PT}\frac{di_P}{dt} \\ u_P &= M_{RP}\frac{di_R}{dt} - M_{PT}\frac{di_T}{dt} + L_P\frac{di_P}{dt} \\ u_R &= L_R\frac{di_R}{dt} - M_{RT}\frac{di_T}{dt} + M_{PR}\frac{di_P}{dt} \end{aligned}$$

Writing Eqn. 6.4 using complex notation for sinusoidal voltages and currents, we obtain

$$\begin{aligned} u_T &= j\omega\, M_{RT}\, i_R - j\omega\, L_T\, i_T + j\omega\, M_{PT}\, i_P \\ u_P &= j\omega\, M_{RP}\, i_R - j\omega\, M_{PT}\, i_T + j\omega\, L_P\, i_P \\ u_R &= j\omega\, L_R\, i_R - j\omega\, M_{RT}\, i_T + j\omega\, M_{PR}\, i_P \end{aligned}$$

To, e.g., obtain the voltage u_P at the probe antenna's terminal, solve for i_T and use $i_P = \frac{u_P}{Z_P}$ and $i_T = \frac{u_T}{Z_T}$:

$$
\begin{aligned}
i_T &= j\omega \frac{M_{RT} i_R + M_{PT} i_P}{Z_T + j\omega L_T} \\
u_P &= \frac{j\omega M_{RP} + \omega^2 \frac{M_{PT} M_{RT}}{Z_T + j\omega L_T}}{1 - j\omega \frac{L_P}{Z_P} - \omega^2 \frac{M_{PT}^2}{Z_P (Z_T + j\omega L_T)}} i_R
\end{aligned}
$$

Alternatively, to express the influence of a varying i_T on the measured voltage u_P for a fixed reader antenna current i_R, one obtains:

$$
u_P = \frac{j\omega M_{RP}}{1 - j\omega \frac{L_P}{Z_P}} i_R - \frac{j\omega M_{PT}}{1 - j\omega \frac{L_P}{Z_P}} i_T
$$

The measured voltage at the probe antenna is hence the (filtered) current in the reader antenna minus the (filtered) current consumed by the transponder. When the transponder draws more current (e.g., due to load modulation), the observed voltage u_P decreases. The two currents are weighted with the respective mutual impedances M_{RP} and M_{PT} and hence essentially $1/d^3$.

Assuming that for eavesdropping the transponder and reader are close, the probe antenna has approximately the same distance to both other antennas. Since the modulation of the reader is stronger (100% ASK) compared to the transponder (approximately 10% ASK)) and since the inductance (and hence the mutual inductances) of the reader antenna is usually larger than that of the transponder, it can be expected that the reader's signal can be received from a larger distance. Indeed, in the following Sect. 6.4.4, we show that the communication R → T can be eavesdropped on from 1.5 to 2.5 times the distance observed for T → R.

6.3.3 Measurement Setup

Passive Eavesdropping To measure the ranges for passive eavesdropping on the communication between reader and transponder, we used a commercial Baltech ID-engine reader [Bal07] configured with a 35 mm x 47 mm antenna specified for read ranges of 4 cm for ISO 14443 and 8 cm for ISO 15693 devices, respectively. The reader was set up to continuously check for the presence of an RFID and subsequently read the transponder's Unique Identifier (UID). The transponder was not fully aligned with the reader's antenna in order to model a realistic situation: one 15 mm x 15 mm corner of the transponder was placed on one corner of the reader antenna. Thus, the coupling between the reader the transponder was relatively weak, still, the reader could in all cases read the UID without errors.

For receiving the communication between reader and transponder, we used a commercially available magnetic antenna made of one 60 cm x 60 cm square loop of copper tubing, cf. Fig. 6.4a. We measured the characteristic impedance of the antenna and found it to be approximately $Z_0\,(2\pi 13.56\,\mathrm{MHz}) = 103\,\Omega + j\,867\,\Omega$ at the relevant frequency of 13.56 MHz. To match this impedance to the standard 50 Ω inputs of the following components and to limit the bandwidth of the received signal to approximately ±1 MHz around 13.56 MHz, we employed a simple L-C filter depicted in Fig. 6.3.

The filtered signal was then amplified by 30 dB using an Anzac AM-110 amplifier (usable for frequencies in the range 0.5–100 MHz) and passed to the USRP2 SDR, cf. Sect. 3.1.1, equipped

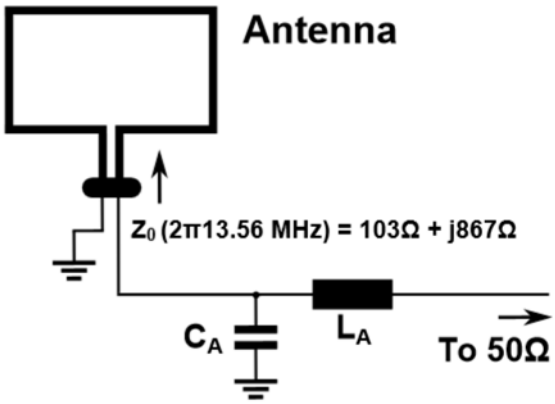
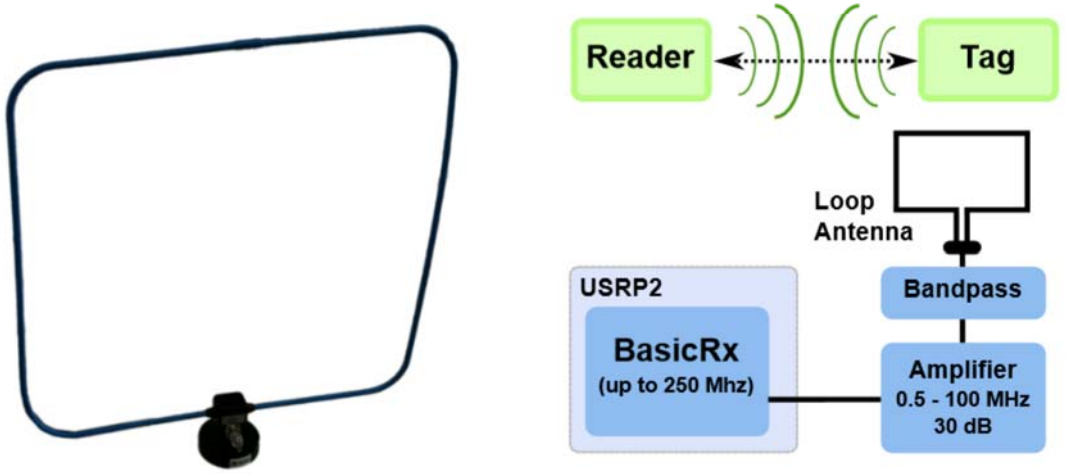


Figure 6.3: Bandpass filter and impedance match to adapt the antenna to a characteristic impedance of 50 Ω at 13.56 MHz

with a BasicRX daughterboard (frequency range 1–250 MHz). The overall structure of the measurement setup for passive eavesdropping is depicted in Fig. 6.4b. The eavesdropping was considered successful if the signal could be demodulated and decoded without bit errors. Note that the (deterministic) response of the transponder (i.e., essentially the UID) was known before and thus, any bit error could be detected without relying on the properties of the protocol's error detection mechanisms, i.e., CRCs and parity bits.



(a) Receive antenna

(b) Block diagram

Figure 6.4: Measurement setup for eavesdropping on 13.56 MHz RFID

Active Communication A similar setup was used for testing the ranges for active communication, however, in this case, the 13.56 MHz field was generated using a self-built power amplifier. The amplifier was designed using the class E design principle [Sok01], i.e., a power MOSFET

switches a load with appropriate filter characteristics to generate the desired sinusoidal output waveform. The schematics of the developed amplifier are depicted in Fig. 6.5.

The OP in the input stage amplifies the 3.3 V square-wave input signal by approximately 3.1 to drive the gate of the IRF510 MOSFET with a voltage of approximately 10 V. The output filter is tuned to present a 15 Ω load to the MOSFET at 13.56 MHz. A 50 Ω load, e.g., a dummy load or our tuned antenna, can the be connected to the output. With a supply voltage of V_{CC} =10 V, we measured a 34 V peak-to-peak voltage at the 50 Ω load, i.e., obtained an output power of $P_{out} = {}^{U^2_{\text{eff}}}/_{50\,\Omega} = {}^{(24.04\,V)^2}/_{50\,\Omega} = 11.56\,W$. We performed all our experiments at this output power in order to avoid damage to the RFIDs when too close to the antenna. With higher output powers, the range could presumably be increased further. To investigate the effect of the antenna, we built two different antennas in addition to the 60 cm x 60 cm square loop mentioned above: a circular loop antenna with three turns of 0.5 mm copper wire and a diameter of 18 cm and a rectangular 3 cm x 5 cm loop antenna on a PCB. All antennas were tuned to 50 Ω at the frequency of 13.56 Mhz.

Then, we repeatedly sent the `REQA` (for ISO 14443) or `Inventory` command (for ISO 15693) and iteratively increased the distance between transponder and antenna until the transponder received insufficient power to answer the request sent by the reader. For generating the input waveform to the amplifier, we employed the GIAnT to output a 13.56 MHz square wave modulated with the appropriate Miller-encoded data.

6.3.4 Practical Results

Using the setup described in Sect. 6.3.3, we determined the ranges for the different test cases (passive eavesdropping, active reading). The results for the three DUT are summarized in Tab. 6.2.

Test case	Range DUT 1	Range DUT 2	Range DUT 3
Eavesdropping T → R	1.85 m	1.60 m	2.85 m
Eavesdropping R → T	4.00 m	4.00 m	4.00 m
Active reading	0.52 m	0.27 m	0.57 m

Table 6.2: Achieved ranges for passive eavesdropping and active reading for DUT 1–3

For eavesdropping, it turned out that the communication R → T could be received from a significantly larger distance compared to the channel T → R. As expected, this result was also independent of a specific transponder and mainly determined by the reader. Due to the relatively weak load-modulation caused by the transponder, receiving the communication T → R was possible from a lesser distance. For example, the data sent by a Mifare DESFire EV1 could be received without errors up to 1.60 m, while the requests of the reader could be picked up from up to 4 m. The highest eavesdropping distance could be achieved for the ISO 15693 transponder DUT 3, where no bit errors occurred up to 2.85 m.

Table 6.3 shows additional details on the eavesdropping experiments. For the “interesting” range regions (i.e., the ones where the first bit errors occur for the DUTs), the signal strength of the channels R → T and T → R and the number of bit errors (average, minimum, and maximum computed over 10 measurements) is given. The signal strength is measured in dBFS,

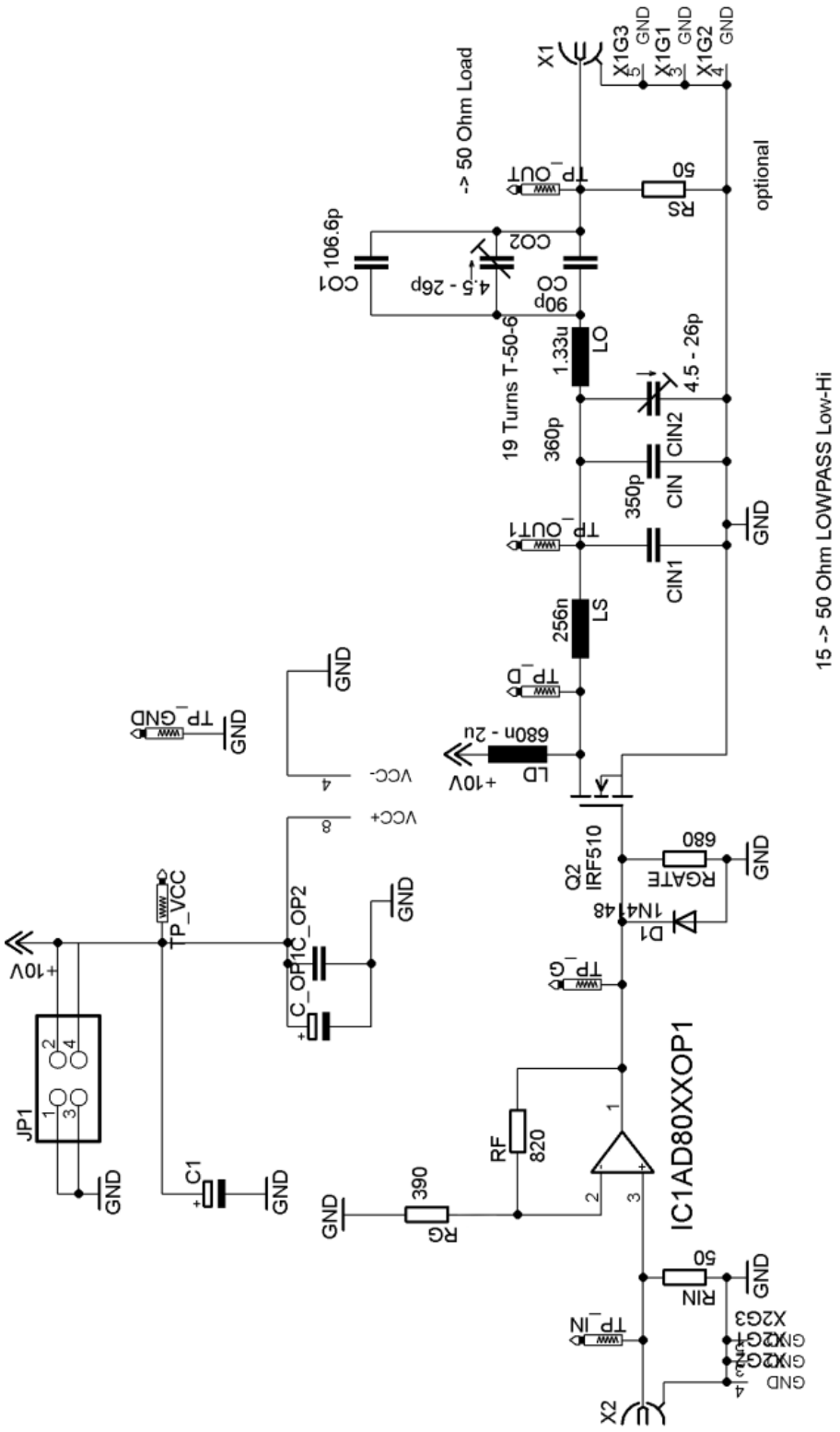


Figure 6.5: Schematics of the employed HF power amplifier

i.e., relative to the maximum input amplitude of the USRP2 of ±1 V. Hence, 0 dbFS correspond to an amplitude of 1 V.

Distance (m)	Channel T → R		Channel R → T	
	Signal strength (dBFS)	Bit errors	Signal strength (dBFS)	Bit errors
DUT 1: ISO 14443, around 2000				
1.80	-69	0/0/0	-47	0/0/0
1.85	-70	0/0/0	-47.5	0/0/0
1.90	-70.5	0.1/0/1	-48	0/0/0
1.95	-70.5	0.7/0/7	-48.5	0/0/0
2.00	-71	0.8/0/4	-49	0/0/0
DUT 2: DESFire EV1, ISO 14443, 2008				
1.50	-68	0/0/0	-43	0/0/0
1.60	-69	0.2/0/1	-45	0/0/0
1.65	-69	2.7/0/16	-45	0/0/0
DUT 3: ISO 15693, around 2008				
2.75	-76	0/0/0	-51	0/0/0
2.85	-76	0/0/0	-51	0/0/0
2.90	-76	10,4/0/52	-51	0/0/0
2.95	-76	7/0/23 3	-51	0/0/0
3.00	-78	10.6/0/23	-51	0/0/0

Table 6.3: Signal levels and average/minimum/maximum number of bit errors for eavesdropping on DUT 1–3

The average difference between the signal strength of the reader and the transponder was found to be -23.92 dbFS, i.e., the reader's signal is stronger by a factor of $10^{23.92/20} = 15.7$. Note that the maximum ranges in Tab. 6.2 are for the case of receiving the signal without errors—a detection whether a transponder is present at all could be performed from a greater distance. To determine the maximum detection range, we found that the signal strength of DUT 1, 2, and 3 exceeded the noise floor up to a distance of 2.20 m, 1.70 m, and 3.00 m, respectively.

For actively reading the transponders, we operated the amplifier at the maximum output power and increased the distance between transponder and antenna until no answer was generated anymore. In this test, the transponder was aligned with the antenna, i.e., moved along the middle axis of the antenna. The results for all DUTs and the three different antennas are summarized in Tab. 6.4.

In all cases, the largest antenna (60 cm x 60 cm square loop) yielded the highest distances. It turned out that modern RFIDs like the Mifare DESFire EV1 have higher energy requirements. While the older ISO 14443 transponder operated at a range of up to 52 cm, the DESFire EV1 stopped working at a distance of 27 cm.

DUT	Range		
	3-turn wire antenna	PCB antenna	1-turn loop antenna
DUT 1	35 cm	19 cm	52 cm
DUT 2	25 cm	15 cm	27 cm
DUT 3	46 cm	23 cm	57 cm

Table 6.4: Achieved ranges for active reading for DUT 1–3

6.4 Active UHF RFID at 433.92 MHz

In contrast to the passive RFIDs analyzed in Sect. 6.3, we focus on an active UHF device[2] in this section. As our DUT, we selected a transponder which is widely used for, e.g., tracking of goods in the supply chain or tagging vehicles, and is employed worldwide both in the commercial and military sector. Our DUT provides more than 128 kB of rewritable memory and communicates employing Frequency Shift Keying (FSK) at a frequency of 433.92 MHz and a bitrate of 27.8 kBit/s at a specified range of 122 m. The energy supply is provided by a 3.6 V lithium battery that, according to the data sheet, lasts for 5 years when issuing two queries per day.

6.4.1 Fundamentals

The UHF interface of the analyzed RFID system is implemented according to ISO 18000-7 [ISO09] and hence uses a binary FSK, i.e., the frequency is increased or decreased by 50 kHz to transmit a 0 or 1, respectively, at a carrier frequency of $f_c = 433.92\,\text{MHz}$. In its default state, a transponder is usually in a sleep state to save power. In order to activate a transponder, a reader has to send a "wake-up" signal by transmitting a constant-frequency signal at 433.92 MHz + 30 kHz for a duration of 2.5...2.7 ms. On receiving this signal, a transponder responds with its UID as the starting point for the further bi-directional data exchange.

The transmitted information is encoded using a Manchester code [Fin03], i.e., the bits are encoded in the transitions of the signal, not in its level. The nominal data rate is 27.8 kBit/s, employing a packet-oriented transmission protocol. Each packet (sent by the reader or the transponder) starts with a synchronization preamble of 1.2 ms, followed by a start bit that also encodes whether the transmission originates from a reader or from a transponder. For error detection and synchronization purposes, each data byte ends with an additional stop bit, and besides, a 16-bit CRC is appended to each packet to ensure data integrity.

6.4.2 Theoretical Background

The Friis transmission equation [Fri46] describes the relationship (in the far-field region) between the transmitted and received power, depending on the distance between two antennas and is denoted in its most basic form as:

$$P_r = P_t\, G_t\, G_r \left(\frac{\lambda}{4\pi d}\right)^2 \tag{6.4}$$

[2]Due to confidentiality reasons, we cannot disclose the product name and the vendor

where P_r denotes the received power, P_t the transmitted power, G_t the gain of the transmitting antenna, G_r the gain of the receiving antenna, λ the wavelength, and d the distance between transmitter and receiver.

Equation 6.4 is important both for the case of passive eavesdropping (where P_r is the power at the output of the eavesdropping antenna) and for actively communicating with the transponder (where P_r is the power received by the DUT). In contrast to the case of passive HF RFIDs (cf. Sect. 6.3.2) for which the H near-field decays $\propto 1/d^3$, the power of the (far-field) EM wave for UHF transmission follows the inverse-square law $\propto 1/d^2$. In this case, the electric and magnetic components of the propagating EM wave both decrease $\propto 1/d$. Hence, much higher operating and eavesdropping ranges are possible for UHF. Additionally, directional antennas with high gain are much more compact due to the relatively short wavelength of 69.09 cm at 433.92 MHz.

6.4.3 Mobile Measurement Setup

For practically evaluating the ranges for eavesdropping in a real-world scenario, all required tools, readers, and transponders need to be portable in order to be transported and autonomously actuated at their destination in the wild or to be operated from our measurement vehicle. The latter is equipped with the mandatory armamentarium, i.e., a ladder and mechanical tools to fix transponders at objects, several lead-acid batteries serving as independent energy supplies, and a 300 W DC-to-AC power converter to provide a 220 V mains supply from the 12 V DC provided by the batteries, as depicted on the top right of Fig. 6.6.

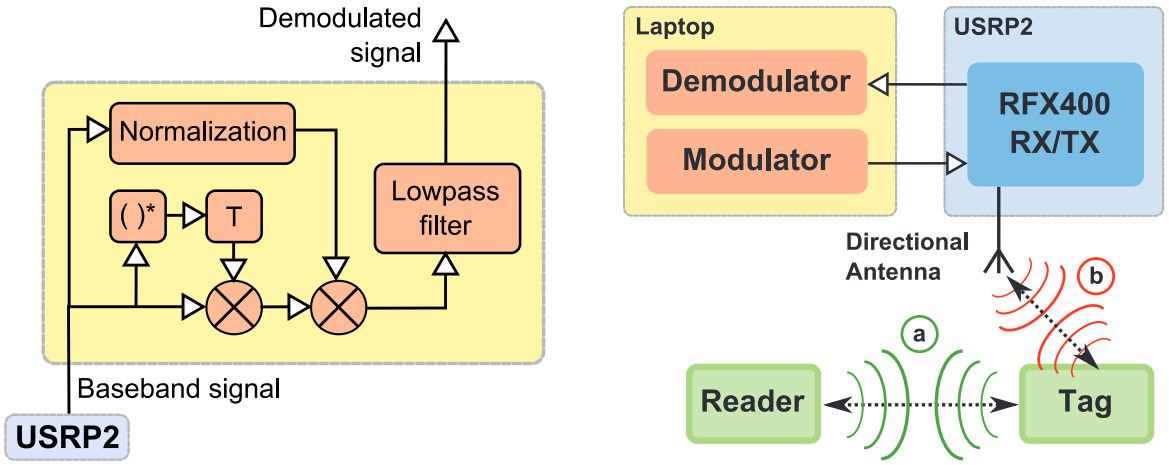
Figure 6.6: Mobile eavesdropping equipment. Left: Yagi-Uda antenna, connected to the data acquisition unit consisting of laptop and USRP2 in the measurement vehicle (right bottom); right top: voltage converter and car battery for the power supply in the wild

For convenience, the measurement vehicle provides an adjustable antenna pole that can be manually steered from the inside to cover an eavesdropping angle of 360° at a tunable antenna elevation of 3.1 to 5 m. The devices for the practical experiments comprise a genuine reader and genuine transponders to be placed at appropriate locations in the wild, as well as a mobile data acquisition unit consisting of a directional antenna connected to the USRP2 (Sect. 3.1.1)

that is controlled by a standard notebook as depicted in Fig. 6.6. An iPhone was employed to record the Global Positioning System (GPS) coordinates of the DUT and each eavesdropping location.

The used Yagi-Uda antenna depicted on the left of Fig. 6.6 is a standard directional antenna (cost approximately 50 EUR) with a length of one meter, suitable for a frequency range of 430–440 MHz. The antenna consists of an array of a dipole driven by the transceiver, a reflector, and six closely coupled director elements that provide a directional gain of $G = 11$ dBi, covering a horizontal angle of 44°. The (this time battery-powered) USRP2 equipped with an RFX400 daughterboard (suitable for the respective frequency range) served for receiving and transmitting arbitrary UHF signals.

The integrated preamplifier of the RFX400 daughterboard was set to a gain of 30 dB throughout our experiments, corresponding to a transmission power of 200 mW. The USRP2 was controlled from a C++ program running on the notebook to enable communication according to Sect. 6.4.1. Our developed software comprises a receiver for the passive detection and eavesdropping of the communication, and can further emulate a genuine reader. Figure 6.7a shows the structure of the employed quadrature amplitude demodulation, where `()*` denotes complex conjugation, $\otimes$ a multiplication, and `T` a delay unit. For actively communicating with transponders, a mode repeatedly transmitting a wake-up command and reading the response was implemented. Figure 6.7b outlines our measurement setup capable to both passively monitor the communication of the DUT with a genuine reader ⓐ and to actively initiate the communication in order to detect and interrogate a transponder when no genuine reader is present ⓑ.



(a) Demodulating UHF signals

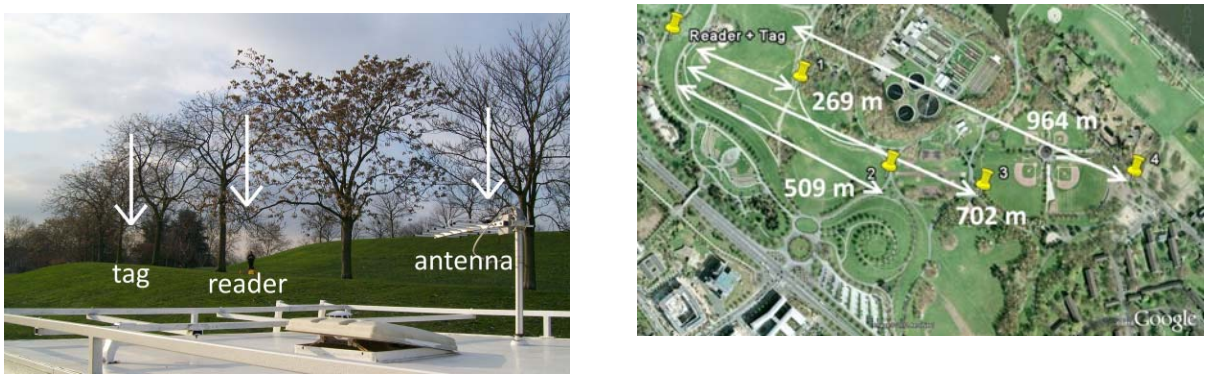
(b) Measurement principle

Figure 6.7: Signal flow graph for demodulating received UHF signals and principle of passive eavesdropping ⓐ and active communication ⓑ

6.4.4 Practical Results

Using the equipment and techniques described in Sect. 6.4.3, this section details on two extensive measurement campaigns carried out at a suitable location in Bonn, Germany, nearby the river Rhine. One campaign aimed at determining the practical range in the context of a passive

attacker, e.g., relevant for eavesdropping, detection and tracking of transponders. The second campaign targeted the range for active communication, as required for active communication in order to detect a transponder or perform a DoS attack. For MITM attacks, the ranges of both approaches have to be taken into account.



(a) Side View (b) Top View

Figure 6.8: Setup and GPS coordinates for passive eavesdropping and detection

For analyzing the eavesdropping range, we attached the DUT to the branch of a tree and placed a genuine reader approximately 15 m away, as illustrated in Fig. 6.8a. The reader was programmed to repeatedly interrogate the DUT during our absence. We then moved away from the DUT and recorded the requests of the reader and the answers of the transponder, increasing the distance in several steps. The exact locations of the measurement positions and their corresponding distances are depicted in Fig. 6.8b on a satellite map from Google Earth. Reaching the end of the areal at a distance of 964 m from the DUT, we were still able to detect the communication with our mobile measurement setup. An exemplary response of the active transponder as acquired during our experiments is shown in Fig. 6.9.

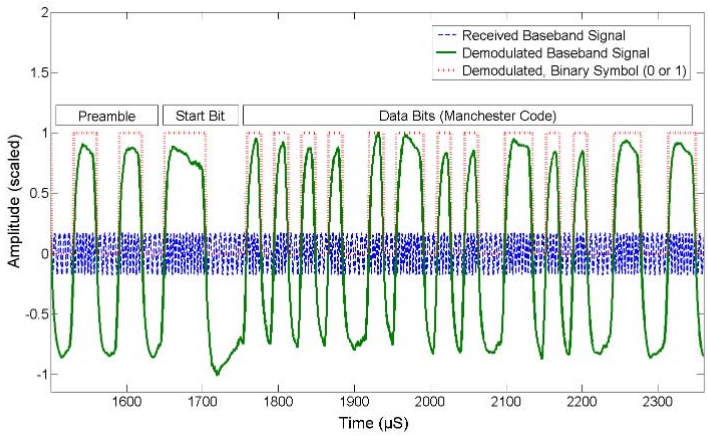
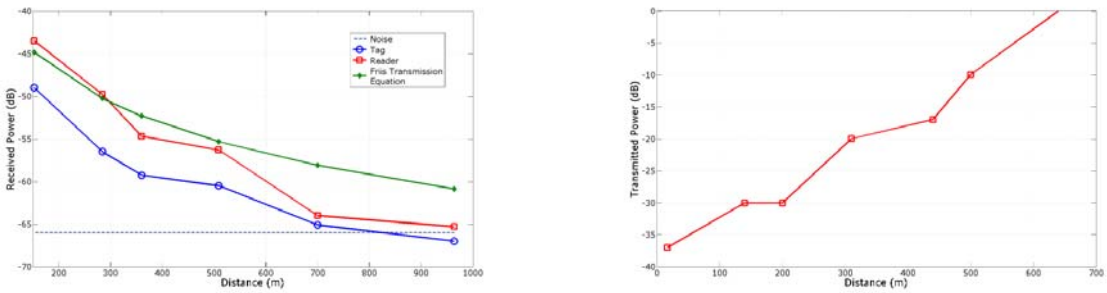


Figure 6.9: Received UHF signal: baseband (blue, dashed), demodulated (green, solid), converted to binary symbols (red, horizontally dashed)

Similarly, we tested the range for active communication by sending wake-up commands and recording the received answers of the DUT. This time, as illustrated in Fig. 6.10, the measurement vehicle remained at a fixed location, while the transponder, being carried by a research assistant, moved away.

Figure 6.10: Setup for active communication

The results for the achieved reading ranges for the case of a passive and an active attacker are shown in Fig. 6.11a and Fig.6.11b, respectively. Concerning the passive approach, the ranges achieved for intercepting the communication from R → T and vice versa were similar, while the reader's signal was slightly easier to detect. For reference, Fig. 6.11a includes the theoretical results obtained when evaluating the Friis equation (Eq. 6.4, Sect. 6.4.1) for the system used in our experiments: the transponder is transmitting with a specified power of 0.6 mW using an approximately isotropic antenna, i.e., $G_t = 1$ (0 dBi). The combined gain of the receiving antenna and the preamplifier is $G_r = 10^{11+30/10}$ (11 dB + 30 dB).



(a) Received power

(b) Active Communication

Figure 6.11: Received power for eavesdropping on T → R (blue, circle) and R → T (red, square) and minimum required transmission power to activate and communicate with a transponder as a function of the distance

Eavesdropping on the communication in both directions was possible from up to 500 m, whereas the eavesdropping was assessed to be successful if each received data packet was correctly decoded (evaluated checking for a correct checksum). Detecting the communication was

successful from a distance of approximately 1000 m. The measured range for the active approach transmitting with a power of 200 mW was also in the order of 500 m.

6.5 Conclusion

In this chapter, we described the relevant security threats in the context of RFID (and similar wireless devices) and showed that the practically achievable ranges for the wireless interface strongly influence many relevant attacks, e.g., eavesdropping, cloning, relay attacks, DoS, RFID malware, as well as tracking and detection of the devices.

We introduced low-cost setups for transmitting and receiving signals both for HF and UHF RFIDs. After reproducing the results of previous studies for HF transponders, we conducted the first real-world analysis of the ranges that can be achieved for an active RFID transponder when passively monitoring communications and actively interrogating the transponder.

For the passive 13.56 MHz RFIDs, we observed asymmetric ranges for eavesdropping on the communication R $\rightarrow$ T and T $\rightarrow$ R: while the reader's requests could be received from 4 m, demodulating and decoding the responses was possible from between 1.60 m and 2.85 m, depending on the technology of the transponder. Actively reading the RFIDs was possible from 0.27 m to 0.57 m, corresponding to a factor of 2.7 to 5.7 compared to the specified range of 0.1 m.

In the case of the active RFID operating at 433.92 MHz, we found that the ranges for eavesdropping on a communication and for passive detection can be increased by a factor of 4 or 8, respectively, compared to the specified operating range (122 m) of the tested active transponder. Similarly, for actively communicating with the transponder, the range could be increased by a factor of 4. Note that the methods and equipment employed correspond to an attacker with a relatively low budget and hence, the ranges have to be regarded as a lower boundary which could be extended by a well-funded adversary.

6.5.1 Implications for HF RFID

Taking the typical application scenarios for HF transponders into account, e.g., proof of identity, payment, and access control, a passively eavesdropping adversary poses a low to medium threat in this context. As shown in Sect. 6.4.4, the attacker has to be within a few meters of the target (e.g., a passport terminal or a payment station) and furthermore is required to use a relatively large antenna, which makes it difficult to perform a concealed attack. Additionally, HF RFIDs often employ some form of cryptographic protection (cf. for instance the Mifare DESFire MF3ICD40 further analyzed in Chap. 7) that protects the exchanged data against eavesdropping.

In contrast, the possibility of actively communicating with a transponder from more than (the specified range of) a few centimeters is potentially more dangerous. For example, a relay attack appears to be a more realistic threat given that the adversary has to be within a radius of approximately half a meter of the victim and does not require almost direct contact.

6.5.2 Implications for UHF RFID

Compared to HF devices, the security risks for the RFID operating at 433.92 MHz are far higher. The devices can be easily cloned from a distance of 500 m, MITM attacks are feasible from afar, and the detection, e.g., to trigger an alarm, is feasible from up to one kilometer. The possible scenarios are manifold, e.g., performing an unauthorized inventory of the warehouse of a competitor, reprogramming transponders attached to containers to change their intended destination, falsify the information recorded by the sensors of the transponder, and so on.

Regarding sleep deprivation attacks, for a conservative estimation, we assume 500 bit to be sent for one interrogation at a bitrate of 20 $^{\text{kBit}}/_{\text{s}}$ (specified bitrate of 27.8 $^{\text{kBit}}/_{\text{s}}$), corresponding to a speed of 40 interrogations per second. Taking a short recovery time into account, and being extremely conservative, we assume one interrogation per second in practice. With respect to the specified lifetime of the transponder's battery, i.e., 5 years at two queries per day, the resulting $5 \times 2 \times 365 = 3560$ queries take about 3560 s—an adversary may hence succeed to empty the battery with a sleep deprivation attack in less than one hour from a distance of 500 m.

The observed ranges also put other eavesdropping attacks, such as those for cloning KeeLoq remote controls, into a new light. Active devices operating at UHF frequencies are very likely as vulnerable as the DUT in this chapter. Accordingly, active wireless technologies should be used with extra care and in the case of security or safety critical environments, other protection measures have to be established, e.g., fences at an appropriate distance and guarded transports of goods.

Chapter 7

Mifare DESFire MF3ICD40

In this chapter, we demonstrate practical, non-invasive side-channel attacks on the Mifare DESFire MF3ICD40 contactless smartcard, a 3DES-based alternative to the cryptanalytically weak Mifare Classic [GKGM+08, NESP08]. We detail on how to recover the complete 112-bit secret key of the employed 3DES algorithm using non-invasive CPA and TA. Our methods can be put into practice with standard equipment, thus posing a severe threat to many real-world applications that employ the DESFire MF3ICD40 smartcard. As an example, we utilize our attacks to extract the secret keys of the Opencard used in Prague and analyze the design of this system. Note that all results in this chapter do not directly apply to the newer AES-based variant DESFire EV1.

Contents of this Chapter

7.1 Introduction

RFID technology has become the basis for numerous large-scale, security-relevant applications, including public transport, wireless payment, access control, or digital identification. The information stored on RFID smartcards, e.g., personal data or a cash balance, is often highly sensitive—however, the access to the air interface and to the device itself is virtually impossible to control. Hence, most modern RFIDs feature cryptographic mechanisms, including encryption and authentication, in order to thwart attacks such as eavesdropping, manipulation, or cloning of a smartcard.

Mifare DESFire MF3ICD40 is a contactless smartcard featuring a cryptographic engine for authentication and encryption based on (Triple-)DES. The smartcard is or was employed in several large payment and public transport systems around the world, e.g., the Czech Opencard [Mag13b], the Australian myki card [Sta11], or the Clippercard used in San Francisco [Met13]. In the course of our research, we also noticed smaller installations, e.g., for

mobile payment or access control, based on the Mifare DESFire MF3ICD40. From a mathematical point of view, the employed 3DES cipher is secure, because no efficient cryptanalytical attacks are known. Thus, we focus on side-channel attacks: using non-invasive and hence non-detectable measurement of the EM emanations of the device, we are able to completely recover any secret 112-bit key stored on the DUT and thus to, for example, read out, manipulate, or duplicate the contents of a Mifare DESFire MF3ICD40 card. We show the practical relevance of our research by applying our attacks to the Czech Opencard and extract secret information, e.g., encrypted files containing personal data and the master key which turned out to be identical for all analyzed Opencards.

The analysis of the Mifare DESFire MF3ICD40 is joint work with Timo Kasper. The main results were published at CHES 2011 [OP11]. The analysis of the Opencard described in Sect. 7.5 was presented in [KOP11b] and in several conference talks [Osw12a, Osw12b].

7.1.1 Related Work

The susceptibility of ciphers running on RFID devices towards SCA was initially shown in [HMF07, PHF08]: the authors present attacks on a known software implementation of the AES executed by a standard, unprotected μC on a self-made prototype RFID, evaluating techniques to overcome problems such as misaligment of the measured signals. With respect to the application of SCA to break commercial, real-world devices, few papers have been published, as most research in this field is carried out by evaluation labs behind closed doors.

Initial results for the black-box analysis of a contactless smartcard are given in a paper published in 2009 [KOP09], proposing the leakage model for RFIDs that forms the basis for our analysis and is outlined in Sect. 2.2.3. However, in 2009 we were unable to recover the complete key and did not disclose to which device our attacks apply. The SCA described in this chapter is thus a significant improvement of [KOP09].

7.1.2 Contribution

In this chapter, we highlight the relevance of SCA in a real scenario by demonstrating the first full key-recovery attack on the popular Mifare DESFire MF3ICD40 smartcard reported in the literature. Doing so, we point out problems and obstacles that occur when conducting SCA in practice. Those issues are often neglected in academic papers. In addition, we present the—to our knowledge—first application of TAs to break cryptographic RFIDs, allowing for potentially very fast determination of the secret key. The remainder of this chapter is structured as follows: We practically analyze the smartcard in Sect. 7.2, detailing on the internal hardware structure of the device. In Sect. 7.3, we extend our findings and present a successful full key-recovery attack on the 3DES engine. After that, in Sect. 7.4, we demonstrate a different approach for obtaining the secret key based on TAs to eavesdrop on the internal databus. We apply the finding to a real-world system in Sect. 7.5. Finally, we conclude in Sect. 7.6, discussing the implications of our results.

7.2 Profiling of Mifare DESFire MF3ICD40

The Mifare DESFire MF3ICD40 [NXP04] is a contactless smartcard initially designed by the semiconductor division of Philips, which became the separate company NXP in 2006. The card is compliant to parts 1-4 of the ISO 14443 A standard. A communication with the card can be performed in plain, with an appended Message Authentication Code (MAC), or with full data encryption using 3DES. The device offers 4 kByte of storage that can be assigned to up to 28 different applications, whereas each application may hold a maximum of 16 files. Depending on the configuration of the access rights, a mutual authentication protocol has to be carried out before accessing the card, ensuring that the symmetric 3DES keys of the card k_C and of the reader k_R are identical.

According to specifications found on the Internet, the smartcard features several functions to thwart physical attacks such as SCA, FI, or reverse-engineering: the IC is built using asynchronous circuits and employs a custom, asynchronous µC design based on the 8051 architectures. Besides, all digital units (i.e., control logic, cryptographic engine etc.) are "intermingled" so that no functional block are discernible, a technology called "glue logic" by the vendor. Note that all results in this chapter do not directly apply to the newer AES-based variant DESFire EV1.

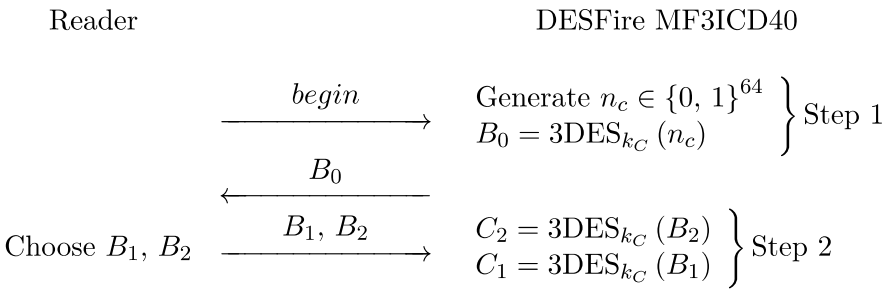


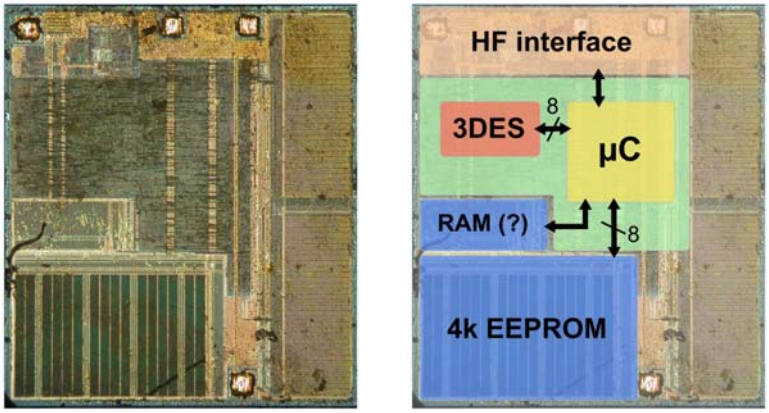
Figure 7.1: Exerpt of the Mifare DESFire authentication protocol relevant for SCA

The authentication protocol of the DESFire MF3ICD40 has been disclosed and can for instance be found in [KvMOP11, Car05]. For the purpose of SCA, we refer to a simplified version given in Fig. 7.1 in the following. $k_C = (k_{C,1}, k_{C,2})$ is the 128-bit 3DES master key (including the parity bits) used by the DUT, whereas the two halves are of size 64 bit each, i.e., $k_{C,1}, k_{C,2} \in \{0, 1\}^{64}$. $\mathrm{3DES}_{k_C}(x) = \mathrm{DES}_{k_{C,1}}\left(\mathrm{DES}^{-1}_{k_{C,2}}\left(\mathrm{DES}_{k_{C,1}}(x)\right)\right)$ denotes a 3DES encryption of a 64-bit value x in Encrypt-Decrypt-Encrypt (EDE) mode. The full command set[1] has been implemented for our custom reader mentioned in Sect. 4.1.

Initially, we were facing a *black-box* scenario, i.e., had (apart from the command set and the specifications in the datasheet) no further knowledge on the inner workings of the device. Hence, profiling to map different portions of a power trace to steps of the operation of the DUT (e.g., a data transfer or an encryption operation) is mandatory before attempting to perform real attacks on cryptographic operations. As a first step, we dismantled the IC, took magnified photographs of the silicon die, cf. Fig. 7.2a, and tried to distinguish the different parts of the

[1] including the necessary commands for changing the key, performing a full authentication etc.

circuit. The hypothetical structure depicted in Fig. 7.2b is a result of this optical inspection and the findings reported in the remainder of this chapter.



(a) IC photo (b) Hypothetical structure

Figure 7.2: The DESFire MF3ICD40 IC

For the profiling and the subsequent analyses, we employed the RFID-specific measurement setup using a full-wave rectifier described in Chap. 4. Note that for all measurements in this chapter, a sample rate of $f_s = 500\,\text{MHz}$ was used. To prepare the actual SCA, we recorded side-channel traces for both steps of the authentication protocol, separately varying either the key of the card k_C or the values for B_1 and B_2 in step 2. To estimate the effect of our analog processing circuitry, we both stored the "raw" signals before demodulation (② in Fig. 4.1 in Sect. 4.1) and the result of the demodulation process (① in Fig. 4.1 in Sect. 4.1).

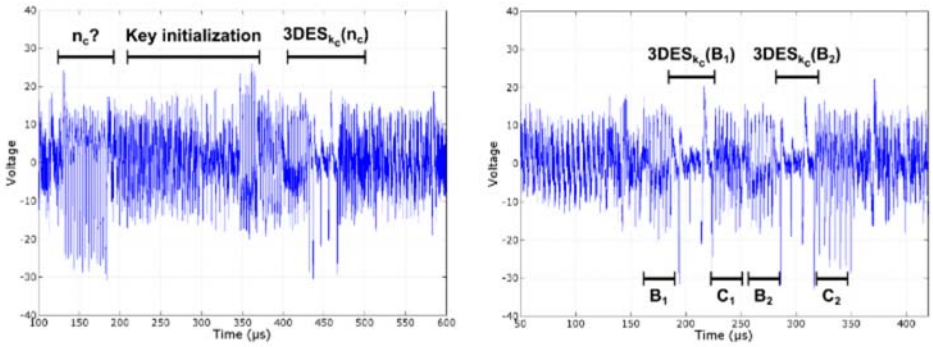
We then performed several CPAs to locate the points in time in the power traces at which the known values for k_C, B_1, and B_2 (and the encryption results C_1, C_2[2]) are processed. Employing an 8-bit HW model, all mentioned values can be precisely pinpointed, cf. Fig. 7.3. We observed a stable value of ≈ 0.15 for the respective correlation coefficient after around 1,000 traces. This suggests that internally, an 8-bit data bus is used to connect the µC to the memory and the cryptographic engine, yielding the structure of Fig. 7.2b. For each byte transferred over this bus, a distinct peak appears in the power trace, whereas the distance between two such peaks indicates an internal bus frequency of $f_{bus} \approx 282.5\,\text{kHz} = {}^{13.56}/_{48}\,\text{MHz}$.

Note that the peaks for data bus transfers later in a trace, e.g. for B_2 or C_2 in Fig. 7.3b, are often *misaligned*, i.e., their exact position slightly varies from execution to execution. The reason for this behavior lies in the non-constant execution time of a 3DES operation, which is further detailed in Sect. 7.3. Hence, it is necessary to re-align the respective parts (for instance, using standard pattern matching approaches) to obtain a significant correlation.

7.3 CPA of the 3DES Engine

Having located the input and output values of the 3DES encryption, we now focus on this part to perform the recovery of the secret key. Fig. 7.4 depicts the part of the trace assumed to repre-

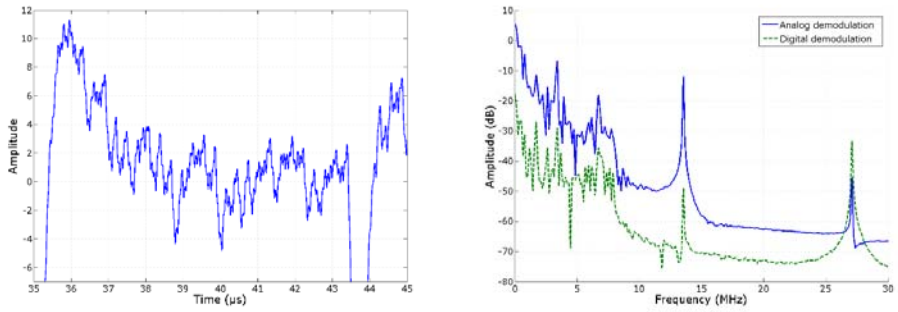
[2]as we know k_C during the profiling phase, we can predict these values that are never output by the DUT



(a) Step 1 (b) Step 2

Figure 7.3: Annotated traces during the authentication protocol (after analog processing)

sent the first (Single-)DES encryption of 3DES_{k_C} (B_2) (Fig. 7.4a) and the associated frequency spectrum (Fig. 7.4b, with (blue, solid) and without (green, dashed) the analog rectifier).



(a) Power profile (b) Spectrum with analog (blue, solid) and digital demodulation (green, dashed)

Figure 7.4: First DES iteration of the 3DES encryption on the Mifare DESFire MF3ICD40

Comparing this part for several traces, we noticed several interesting properties: first, the amplitude of the traces is significantly lower during the supposed encryption, which coincides with the statements in the available DESFire documentation that a dedicated low-power hardware engine performs the cryptographic operation. Second, the length of one DES operation varies from execution to execution, even if the input data and the key are kept constant. This hints at a countermeasure based on randomization in time being employed to thwart CPA.

In order to further investigate this issue, we analyzed the timing characteristics of the part presumably belonging to the first DES operation. We determined the duration between the first and the second large peak (first and second vertical, dashed line in Fig. 7.5) and extracted the local maxima of the respective part (green circles in Fig. 7.5) for 100,000 traces.

We statistically evaluated the measured durations and the number of local maxima. Figure 7.6a depicts the histogram for the approximate durations of the first DES operation: one iteration takes 8.2 µs on average. This duration varies in discrete steps of 290 ns over a total

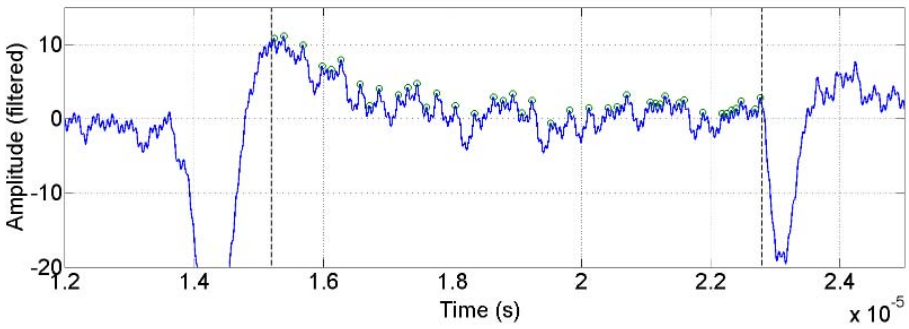
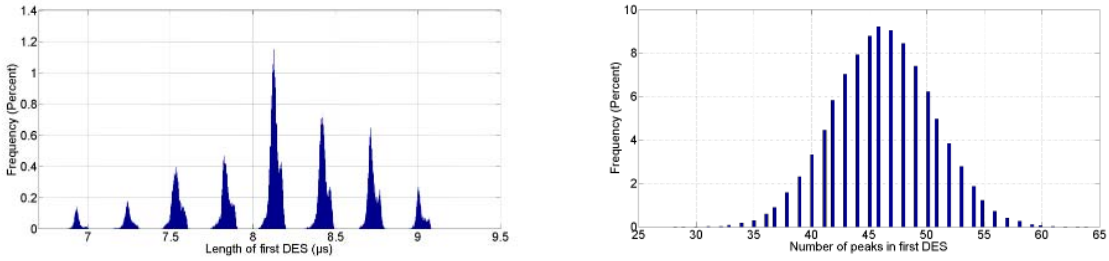


Figure 7.5: Timing analysis of a DES operation. Vertical, dashed lines: start and end point of DES. Green circles: Extracted local maxima of the trace

range from 6.9 µs to 9.1 µs. This suggests that the cryptographic engine executes up to eight ($\lceil (9.1-6.9)/0.29 \rceil$) "dummy" rounds (and four on average) based on an internal Random Number Generator (RNG) to impede SCA. Taking into account the distribution of the number of local maxima (i.e., an approximation for the number of clock cycles), we observed an average of 46.2 maxima (with a minimum of 28 and a maximum of 64). For an iterative implementation of the DES with 16 rounds, this would correspond to approximately two to four peaks per round (with an average of three peaks per round).



(a) Duration

(b) Number of peaks

Figure 7.6: Histograms of the duration and number of peaks for the first DES encryption

7.3.1 Prelimary Analysis

To prepare the actual key-recovery, we first attempted to characterize the leakage of the 3DES engine. We found a suitable power model by correlating with the full intermediate 64-bit states, i.e.,

$$\left(L_i^{(n)}, R_i^{(n)}\right), 0 \leq i \leq 16, n \in \{1, 2, 3\}$$

where n denotes the Single-DES iteration within the complete 3DES using a known key, for details cf. [NISc]. Conducting several experiments, we found the HD model to yield a significant

correlation and were able to locate the first few rounds of the DES, as depicted in Fig. 7.7 for rounds 0→1, 5→6, 10→11, and 0→1 of the second DES iteration.

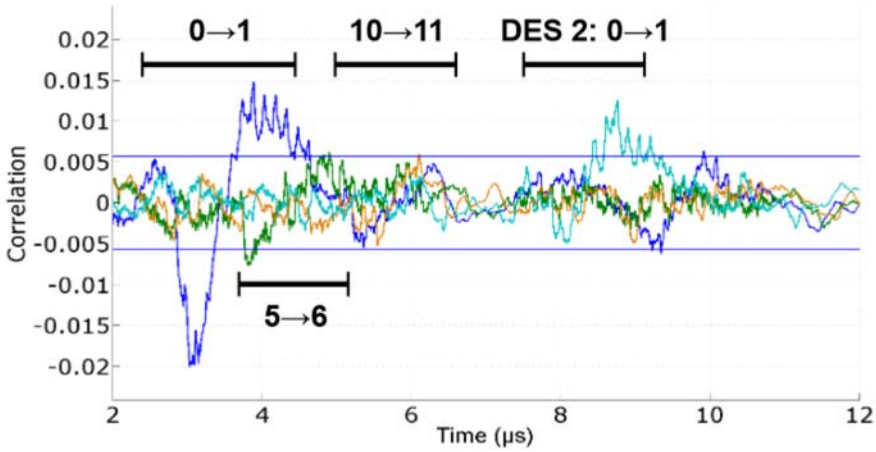


Figure 7.7: Correlation coefficients in the time domain for the HDs between rounds of the 3DES, 500,000 traces

However, as evident in Fig. 7.7, this approach only is able to locate the first few rounds (with decreasing correlation), supposedly due to the randomization mentioned above. To solve this problem, we tried out methods to overcome misalignment suggested in the literature, including comb filtering or windowing [CCD00], DTW [vWWB11], and SCA in the frequency domain or DFA. Our results show DFA to yield the best overall correlation, using the steps described in Sect. 2.4.2.

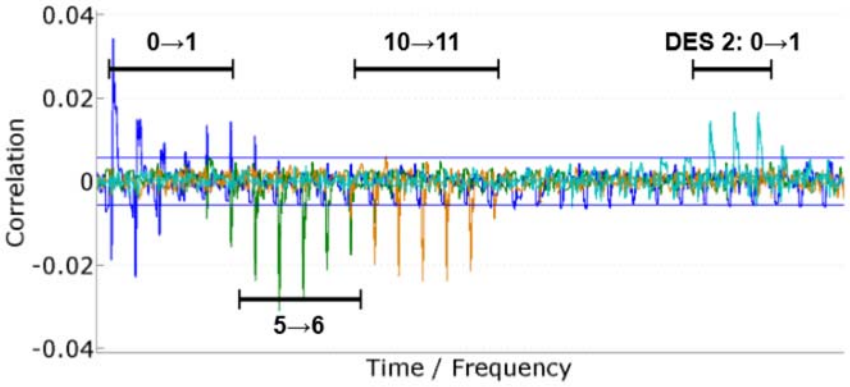


Figure 7.8: Correlation coefficients in the frequency domain for the HDs between rounds of the 3DES, 500,000 traces

The optimal value for the size of each window in the time domain was determined to be $w_{in} = 1.5\,\mu s$, with an overlap of 75 % between adjacent windows. The strongest leakage occurs for low frequencies, hence, we limited the analyzed spectral range to 0...16 MHz. Fig. 7.8 shows

the according correlation coefficients for the respective rounds of the cipher—in contrast to the analysis in the time domain, all rounds are clearly distinguishable.

In order to quantify the improvement caused by the employed analog and digital processing methods, we compared the maximum correlation coefficient over the number of traces for the 32-bit HD $R_0 \rightarrow R_1$ (again using a known key), with a detailed plot of the respective values given in Fig. 7.9.

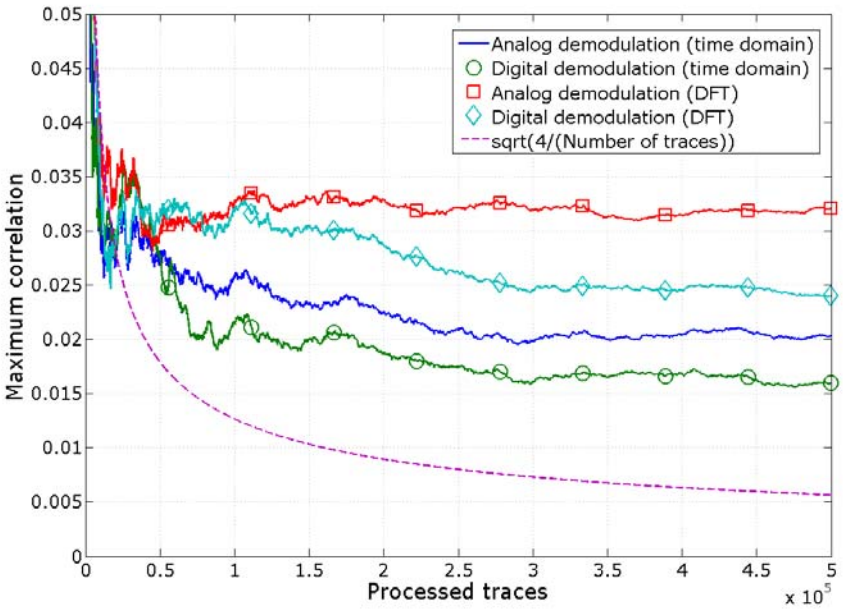


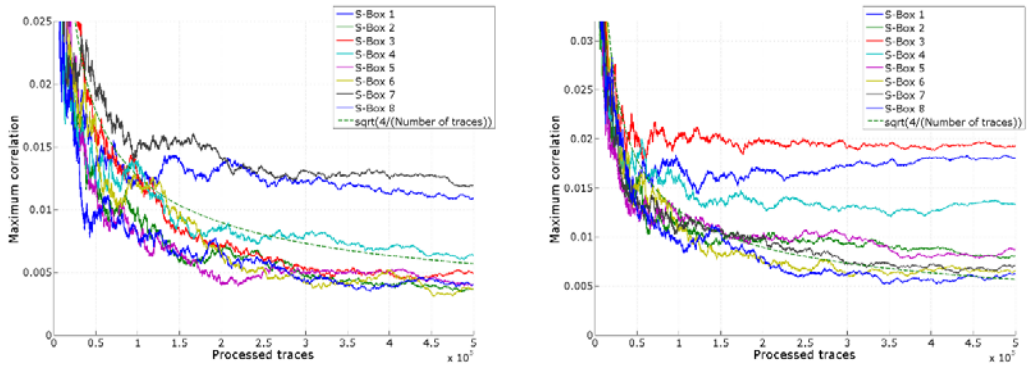
Figure 7.9: Maximum correlation coefficient (32-bit HD $R_0 \rightarrow R_1$) for (a) analog demodulation (blue, solid), (b) digital demodulation (green, circle), (c) analog demodulation with DFA (red, square), (d) digital demodulation with DFA (cyan, diamond)

In all cases, the correlation converged rather quickly to a significant value far greater than $4/\sqrt{\#\,\text{traces}}$ (cf. Sect. 2.3.2), yet, a distinct gain due to both analog and digital processing is discernible: while the digitally demodulated traces without re-alignment by DFA result in a stable value of ≈ 0.015, the combination of analog demodulation with DFA yields ≈ 0.032, that is, an improvement by a factor of two. As a result, we utilized these pre-processing techniques for the full key-recovery presented in Sect. 7.3.2, taking the fact into account that for an actual attack, we have to target each 4-bit S-box output separately, so smaller overall correlations are to be expected.

7.3.2 Full Key-Recovery

Based on the findings of the profiling phase, a CPA can be mounted to obtain the full 3DES key by recovering the 6-bit part of the round key for each S-box, starting with the first round of the first DES. To make use of all available information, a natural choice is to target the full 4-bit output of each S-box in the HD $R_0 \rightarrow R_1$. However, for the case of the DESFire MF3ICD40, this turned out to be problematic: Fig. 7.10 shows the maximum correlation coefficients for

the correct key candidate for a standard CPA in the time domain and DFA in the frequency domain, respectively.



(a) Time domain

(b) Frequency domain

Figure 7.10: Maximum correlation coefficient for the correct key, 4-bit model, HD $R_0 \rightarrow R_1$ for all S-boxes

Although the complete key is discernible after $\approx$ 450,000 traces in Fig. 7.10b, the stable value for the correlation significantly differs depending on the S-box, causing the attack to fail for five S-boxes when performed without re-alignment by means of DFA, cf. Fig. 7.10a. Testing other prediction functions, a single-bit CPA (which is equivalent to the classic DPA) proved to be the most successful approach. As depicted in Fig. 7.11, for each S-box there is at least one bit providing sufficient leakage to allow our attack to succeed after approximately 250,000 traces and 350,000 traces with and without DFA, respectively.

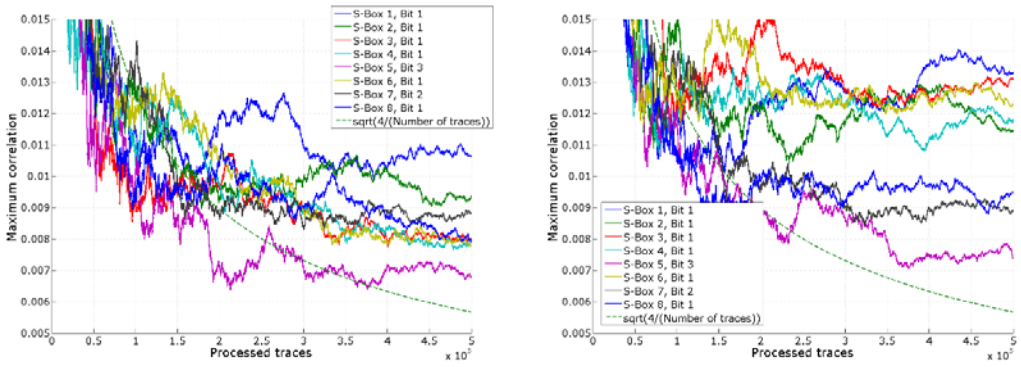
For the sake of optical clarity, the maximum correlation for wrong key candidates has been omitted in the above figures. Yet, we performed the actual key-recovery computing these correlations as well and verified that in all cases, the correlation for the wrong candidates was below $4/\sqrt{\#\,\text{traces}}$, i.e., there were no "ghost peaks" that might interfere with the retrieval of the correct key. Besides, the results are not limited to the first round of the first DES: the analysis equivalently works for other rounds of the first DES (to recover the remaining eight bit of $k_{C,1}$) and for the second DES iteration[3] (to obtain $k_{C,2}$).

Note that the attack on the full 3DES is carried out in separate steps and can tolerate some unknown key bits for the CPAs on the different DES iterations. More precisely, to recover the full 112-bit key, the following sub-attacks are used:

(1) Perform a CPA on the first round of the first DES encryption to recover 48 bit of the 56-bit key $k_{C,1}$ using 250,000 traces.

(2) Perform a CPA on the output of the first DES for the remaining unknown 8 bit of $k_{C,1}$ (i.e., 256 candidates). Since the full 64-bit state can be predicted, 150,000 traces are sufficient for this step.

[3] in this case, alignment to the start pattern of this operation is necessary



(a) Time domain

(b) Frequency domain

Figure 7.11: Maximum correlation coefficient for the correct key, 1-bit model, HD $R_0 \rightarrow R_1$ for all S-boxes

(3) Perform a CPA on the first round of the second DES decryption to recover 48 bit of the 56-bit key $k_{C,2}$ using 250,000 traces.

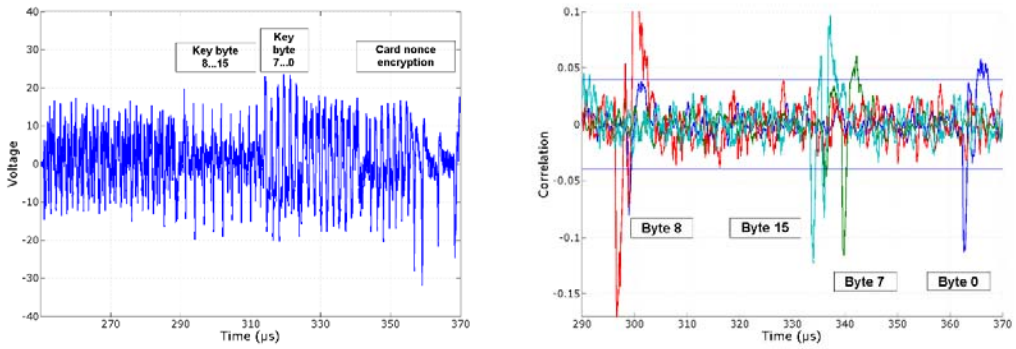
(4) Finally, recover the remaining 8 bit of $k_{C,2}$ by predicting the result of the complete 3DES using 256 candidates. Here, due to the strong leakage of the data bus (cf. Sect. 7.2), less than 1,000 traces are sufficient.

In summary, as a result of this section, we conclude that the extraction of the complete secret 3DES key from a Mifare DESFire MF3ICD40 can be carried out with approximately 250,000 traces, which can be collected in approximately seven hours using our current measurement setup.

7.4 Template Attack on the Key Transfer

As observed during the profiling phase described in Sect. 7.2, the internal databus of the DUT seems to be completely unprotected and exhibits a far stronger HW leakage than the cryptographic engine analyzed in Sect. 7.3. Thus, TAs to obtain information on internal values transferred over this bus can be expected to work with a far lower number of traces compared to a CPA. Of special interest is the initialization of the cryptographic engine before the start of the actual 3DES operation: our analysis shows that the transfer of the secret key can be identified in the EM trace after the reader has sent the initial `begin` command in the authentication protocol (that is, during Step 1 in Fig. 7.1).

Figure 7.12a depicts a trace for the loading of the key and indicates the internal order of operation: by repeatedly changing the key and performing a CPA using the HW of each key byte, we found out that the 3DES key is initialized in two steps. First, the upper eight bytes ($k_{C,2}$) are transferred, starting with the Least Significant Byte (LSByte). After that, the lower half $k_{C,1}$ (i.e., byte 0 ... 7) is transmitted, this time in reverse byte order. In both cases, the (redundant) parity bits are not removed prior to the key transfer, suggesting that they are discarded internally by the cryptographic engine. Fig. 7.12b exemplarily shows the corresponding



(a) Side-channel trace

(b) Correlation

Figure 7.12: Transfer of the 3DES key over the internal databus

correlation peaks for the key bytes 0 (blue), 7 (green), 8 (red), and 15 (cyan), allowing to exactly pinpoint the time instants at which information on a specific byte is leaking.

To systematically evaluate the success rate of TAs for the case of the transfer of the key on the Mifare DESFire MF3ICD40, we obtained 8,000 traces for each possible value of a targeted key byte[4]. Here, we only address byte 0 and 15, however, our results hold for all other bytes as well. 4,000 traces are used for the training set, while the other 4,000 form the test set—in total, to cover all 256 possible values for a byte, we acquired $2 \cdot 256 \cdot 4{,}000 = 2{,}048{,}000$ traces. Again, we also compared the quality of analog demodulation compared to its digital equivalent and hence recorded traces both before and after the analog circuitry. Let $\mathcal{S}_b^{training} = \{\vec{x}_{b,0}, \ldots, \vec{x}_{b,3999}\}$ be the training set and $\mathcal{S}_b^{test} = \{\vec{x}_{b,4000}, \ldots, \vec{x}_{b,7999}\}$ the test set, where $\vec{x}_{b,n}$ denotes the n'th trace for a specific byte value $0 \leq b < 256$.

We then used the evaluation methods described in Sect. 2.3.3 to compute the most likely candidate for each test set $\mathcal{S}_b^{test}$, $0 \leq b \leq 255$. Table 7.1 summarizes the results of our analysis both with (Tab. 7.1a) and without analog pre-processing (Tab. 7.1b).

The average bit error rates were estimated by applying Alg. 2 (Sect. 2.3.3) for each byte, using the corresponding test set $\mathcal{S}_b^{test}$ and computing the HD between the detected and the actual value b. The rank denotes the position of the correct key candidate when comparing a test set $\mathcal{S}_b^{test}$ to the training set and sorting the resulting 256 distances smallest-first. Thus, a rank of 1 means that the correct key candidate is identified as the most likely value, i.e., has a minimal distance to the correct template in the training set.

Note that the average rank of Tab. 7.1 is somewhat misleading, as most candidates are correctly identified and lead to a rank of 1. Figure 7.13 shows a histogram for the number of key candidates yielding a specific rank. For key byte 15 (Fig. 7.13b), approximately 70% of all candidates are recovered correctly with rank 1. For key byte 0 (Fig. 7.13a), 29% of the candidates reach rank 1, 14.4% rank 2, and 16% rank 3.

The upper half $k_{C,2}$ can be recovered with significantly less error than $k_{C,1}$, which interestingly admits a rather different leakage characteristic. In either case, the remaining uncertainty can

[4] the training and test sets were acquired in separate measurement campaigns to rule out effects due to slightly varying environmental conditions

Key byte	Distance	Bit errors	Rank (Average/Min/Max)
0 ($k_{C,1}$)	DiffMeans	2.04	14.29 / 1 / 228
	Euclidean	2.06	14 / 1 / 236
	Mahalanobis	**1.77**	9.89 / 1 / 224
15 ($k_{C,2}$)	DiffMeans	0.55	3.86 / 1 / 194
	Euclidean	**0.51**	3.91 / 1 / 198
	Mahalanobis	0.64	3.96 / 1 / 211

(a) With analog processing

Key byte	Distance	Bit errors	Rank (Average/Min/Max)
0 ($k_{C,1}$)	DiffMeans	2.33	22.03 / 1 / 231
	Euclidean	2.32	21.36 / 1 / 228
	Mahalanobis	**2.26**	25.49 / 1 / 228
15 ($k_{C,2}$)	DiffMeans	0.71	3.66 / 1 / 166
	Euclidean	**0.70**	3.66 / 175
	Mahalanobis	1.22	9.76 / 239

(b) Without analog processing

Table 7.1: Average bit error rates and rank of the correct key candidate for the key recovery based on templates (using 4,000 traces per template)

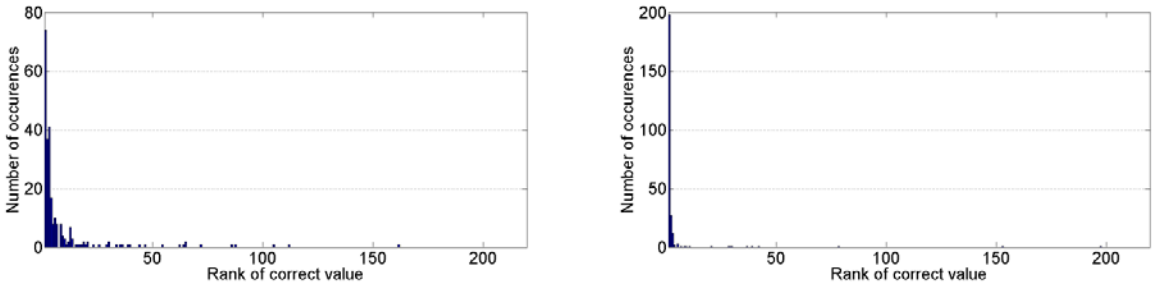
be accounted for using exhaustive search over the key candidates, starting with the ones having the smallest distance to the training set. An algorithm shown to be optimal for this case can be found in [VCGRS13].

Besides, note that the Least Significant Bit (LSB) of each key byte is (unused) parity information which could, for creating an identical clone, be recovered using the `GetKeyVersion` command of the DESFire MF3ICD40. Also, the TA could be combined with the CPA of Sect. 7.3.2. In the simplest case, the CPA could be used as in Sect. 7.3.2 to recover the key of the first and third DES iteration separately. The correct key (determined with a TA) of the first DES can be checked with a CPA (predicting the full 64-bit state), for which approximately 150,000 traces would suffice.

Limitations

Compared to the CPA presented in Sect. 7.3, the key recovery by means of templates might be carried out with far less traces and hence within a very short time—in our current setup, recording 4,000 traces is a matter of minutes—thus potentially posing a severe security threat in a scenario in which an adversary either has to extract many different keys (e.g., due to a key diversification mechanism, cf. Sect. 7.5) or faces a constant risk of being detected.

However, due to the necessity for a profiling phase, implementing the approach in practice turns out to be highly problematic as already mentioned in Sect. 2.3.3. For the results given in Tab. 7.1, we could employ the same DUT, whereas in a real-world attack, the profiling and the attack device are different. In our experiments with different cards, we observed significantly



(a) Byte 0, Mahalanobis (b) Byte 15, Euclidean

Figure 7.13: Histogram of the rank of the correct key candidate in a TA on key byte 0 and 15 for traces with analog processing

differing leakage characteristics, even if the measurement setup (i.e., the positions of the EM probe and the DUT on the antenna) was kept exactly fixed.

7.5 Analysis of a Real-World System: The Opencard

To verify the efforts required for extracting all keys of a commercial system, we analyzed the Opencard system deployed in Prague [Mag13b]. In the multi-purpose Opencard application, Mifare DESfire MF3ICD40 cards enable, amongst others, ticketing for the public transport, usage of the public library, and payments for parking in the city center. As of December 2010, there were approximately 600,000 cards in the field [The10], and the initial budget to realize the system was approximately 800 million CZK [Jon12], i.e., approximately 31 million EUR (using the exchange rate of April 2010). On their website [Mag13a], the system operator states that

> "Data on the Opencard chip is secured by encryption, and by other security mechanisms, it also requires an access code. It is not possible for anyone without the access code to read personal data from the chip, even with the help of a chip reader. Access codes are strictly protected. The strict security precautions prevent leakage of this data. [...] Only the service "Transport Check-in System" enables access to the information about a holder's date of birth in order to process the discounts based on age. Therefore there is only a very minimal chance that the data would be abused."

In order to investigate whether this statement is true and how secure this example of a real-world system using Mifare DESFire MF3ICD40 is, we obtained several Opencards. In our initial analysis, we noted that the access to the application list of an Opencard is not restricted, i.e., this information can be read without any authentication. We found three applications with the hexadecimal IDs `50 11 f2`, `50 82 f1`, and `ff ff ff`.

From our experiments, we deduced that very likely, application `50 11 f2` is dedicated to public transport (thus termed `AppTransport` for the remainder of this chapter), application `50 82 f1` to parking (`AppParking`), and the last application `ff ff ff` (`AppAdmin`) to store general (administrative) information such as the expiry date of the card.

(a) Card (b) Card in measurement setup

Figure 7.14: Analyzed real-world system: The Opencard

Applying the CPA of Sect. 7.3.2, we extracted the secret keys of all applications and read out the (decrypted) content of several Opencards. We found that the master key used for, e.g., deleting a complete card ("Format" command), seems to be identical for all cards. Likewise, the keys for `AppAdmin` were the same for all analyzed cards.

`AppAdmin` contains three files, with the first being a value file that seems to store a version number of the system. The remaining two files are standard data files that—apart from data with unclear meaning—store the expiry and issuance date, and, for personalized cards, the date of birth of the owner. The following dump exemplarily shows (partially anonymized with `xx`) the content of this application for an unpersonalized Opencard:

```
FileID = 0:
Value = 03 00 00 00 (version information)

FileID = 1:
89 80 04 52 09 xx xx xx e1 b9 65 xx 34 d0 xx xx xx 25 8c 5b bb xx 6a
64 57 86 70 55 xx xx xx 3f e7 b8 xx 0c 73 xx xx xx 10 da 87 8a xx 3f
48 43 ab 1c 80 00 00 00 00 00

FileID = 2:
02 41 33 22 7b xx xx xx 16 01 41 00 00 00 00 00 10 20 11 16 14 20 11
30 33 94 8c 5f xx xx xx 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 23 53 80 00 00 00 00 00
```

The issuance and expiry date is stored in file 2 beginning at byte 16, whereas two decimal digits of the date are encoded as two hexadecimal digits. Thus, the present card was issued on Nov 16th, 2010 (`10 20 11 16`) and expires on Nov 30th, 2014 (`14 20 11 30`), dates that are consistent with the dates printed on the card. For a personalized Opencard, the date of birth of the owner is stored in file 1, with a format similar to the one for the expiry and issuance date. For example, the date of birth "Dec 24th, 1984" would be encoded as `84 19 12 24`.

`AppTransport` contains one 480-byte backup data file. In contrast to `AppAdmin`, each card has individual, diversified keys for this application. These keys seem to be derived based on the UID of the card: we copied the complete content (including all keys) of one card with a valid public transport ticket to a blank card. However, this card was not accepted as valid by the ticketing terminals. Thus, the keys and/or the data stored in `AppTransport` ensure that a transport pass is bound to one specific card. The following dump shows (again partially anonymized) the content of the data file in `AppTransport` (leaving out final zero bytes):

```
FileId = 0:
01 4d 00 00 00 00 00 00 00 20 xx xx 13 8d 2d bc 8d 2a 37 90 xx 36 02
19 00 xx d9 7b ff 28 77 ae c7 xx xx 42 9d 61 76 bd 44 59 69 xx e6 e2
b2 b9 xx 02 19 00 0e ef 87 6d xx xx 15 10 73 15 9b 3a e7 f1 xx 2b 04
71 91 xx 42 92 dc 1d 1b
```

`AppParking` stores three (data and backup data) files that are—similar to `AppTransport`—secured with diversified keys that seem to be individually derived for each card. However, we found that the balance for paying for parking is stored in plain in the third byte of file 1. We charged a card to 200 CZK, leading to a value of `0xC8` = 200 at the respective position. Then, we payed 10 CZK and read the file again. This time, we observed a value of `0xBE` = 190 for this byte. The following (again partially anonymized) dump was taken after this payment operation:

```
File ID = 8:
01 00 00 00 00 01 00 40 00 40 xx xx xx 10 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 59
02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
... three lines of zero bytes ...
00 00 00 00 00 00 00 00 00 00 00 00 00 15 xx xx e4 00 00 00 00 0f 7c
79 80 00 00 00 00 00

File ID = 0:
02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 xx xx xx 06 2e b2 20
00 f6 xx xx 55 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 87 xx xx ba 00
00 00 00 1f xx 0c 80 00 00 00 00 00

File ID = 1:
01 00 be 00 01 xx xx 00 01 00 00 00 00 58 00 00 00 00 00 00 00 00 58
00 00 00 00 00 00 00 00 58 00 00 00 00 00 00 00 00 58 00 00 00 00 00
00 00 00 58 00 00 00 00 00 00 00 00 58 00 00 00 00 00 00 00 00 58 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 xx xx xx a8 00
00 00 00 15 60 c0 80 00 00 00 00 00
```

As mentioned, experiments attempting to copy the data and keys from one Opencard to a blank card with a different UID were unsuccessful. However, using an RFID emulator, e.g., the open-source hardware Chameleon [KvMOP11], an exact duplicate of an Opencard can be created

using the results of our analysis. In reaction to the weaknesses of the DESFire MF3ICD40, the system operator is currently migrating to DESFire EV1. Opencards we obtained in February 2013 are already based on this more secure IC.

For comparison, we also briefly analyzed the Clippercard employed in San Francisco [Met13]. The cards we obtained in July 2011 were as well based on the DESFire MF3ICD40, however, reading the data on the card was not requiring an authentication. There is even an Android app "FareBot" [But13] that, amongst others, can read and decode the data on the Clippercard. However, the verification of the card's authenticity by a terminal is performed using the DESFire authentication mechanisms. Besides, the DESFire is configured to append a MAC to every (plaintext) message. Similar to the Opencard, the keys seem to be diversified based on the UID—the analyzed cards used different keys.

In summary, it can be said that even the DESFire MF3ICD40-based Opencard system is relatively secure. Due to the diversified keys, an adversary has to extract numerous keys (with approximately seven hours of measurement per key, cf. Sect. 7.3.2) to fully read one specific card. Even then, the adversary can only clone exactly this card and furthermore only when using an emulator. The main weakness of the system as analyzed here is that some applications lack diversified keys—which is especially a problem in case of the file in `AppAdmin` storing the owner's date of birth. Besides, the identical master key allows an adversary to delete applications, e.g., in a DoS scenario or to obtain a blank card with the official Opencard print.

7.6 Conclusion

We demonstrated several SCA attacks to fully recover the 3DES key of the Mifare DESFire MF3ICD40, employing standard equipment in an academic measurement setup that can be built for approximately USD 3000. As we figured out the details of the implementation of the DUT, the attacks can be realized within a few hours (to collect approximately 250,000 traces for a CPA), and hence pose a severe threat to the security of DESFire-based real-world systems.

System integrators should be aware of the new security risks that arise from the presented attacks and can no longer rely on the security of the used 3DES cipher of the DESFire MF3ICD40. Hence, in order to avoid, e.g., manipulation or cloning of smartcards used in payment or access control solutions, proper actions have to be taken: on the one hand, multi-level countermeasures in the backend allow to minimize the threat even if the underlying RFID platform is insecure, cf. Sect. 12.5. For long-term security and when developing new systems, we recommend to use certified smartcards, for instance, the AES-based Mifare DESFire EV1, which passed an EAL-4+ evaluation [BSI08] and which comprises SCA countermeasures that thwart the analysis methods presented in this chapter.

Chapter 8

SimonsVoss Access Control System

In the past years, various electronic access control systems have been found to be insecure. One of remaining, so far presumably secure electronic locking systems, the SimonsVoss system 3060 "G2", is installed in numerous buildings and objects around the world. However, the system is built using undisclosed and proprietary cryptographic primitives, and no information on the exact functionality is available. To estimate the level of security provided by the SimonsVoss system, we reverse-engineered the details of the authentication protocol carried out between a transponder and the counterpart in the door. Apart from mathematical weaknesses that allow an adversary to impersonate a specific transponder given access to the door only [SDK+13], we discovered that a system-wide master secret used for deriving a transponder's key is stored on each lock within an installation. We show that an adversary can extract this master key using a non-invasive EM side-channel attack given 15 minutes of physical access to the electronics of an arbitrary lock. Possessing the master key, any transponder can be emulated, ultimately giving the adversary access to all doors secured with the SimonsVoss system. As a general outcome of this chapter, we conclude that proprietary, obscurity-based mechanisms—as used by SimonsVoss—only protect against SCA (and cryptanalysis in general) as long as the details remain secret. This can lead to a false sense of security, since numerous techniques are available to reverse-engineer such vendor-specific functionality.

Contents of this Chapter

8.1 Introduction

Electronic access control systems have become popular and are on the way to replace conventional mechanical locks and keys especially in business applications. Often, the security of these systems is based on keeping the proprietary protocols and cryptographic algorithms secret. Admittedly, this approach often prevents both mathematical analysis and implementation

attacks as long as the details remain undisclosed. We focus on the latest generation "G2" of the widespread digital locking and access control system 3060 manufactured by SimonsVoss Technologies AG. A transponder, the digital equivalent of a mechanical key, wirelessly communicates with an electronically enhanced locking cylinder that—after successful authentication—mechanically connects the otherwise freewheeling knob to the bolt so that the door can be opened. SimonsVoss is the European leader in digital locking and access control systems [Sima] and has sold over three million transponders for more than one million electronic locks. Most of the over 10,000 large installations worldwide are high-value (and/or high-security) targets. Despite the high price of the digital locks compared to their mechanical counterparts, the purchase can pay out due to the flexible administration of access permissions via a wireless link, especially in buildings with a lot of doors and users.

The attack described in this chapter is the result of joint work with Daehyun Strobel, Benedikt Driessen, Gregor Leander, Timo Kasper, and Falk Schellenberg. The outcomes of the reverse-engineering and the mathematical analysis will appear at CRYPTO 2013 [SDK+13]. The side-channel attack will be published at SAC 2013 [OSS+13].

8.1.1 Related Work

In the context of conventional mechanical locks, well-known techniques can be used to duplicate individual keys when physical access is given. Likewise, the security of mechanical pin tumbler locks can be bypassed, e.g., by means of lockpicking with special tools. The practical threat of these attacks is usually negligible as long as each key and each lock has to be treated individually. When it comes to the security of large installations and the corresponding master keys, even in the mechanical world the impact becomes more severe: In [Bla03], a procedure is described that allows the creation of a master key to open all doors in an installation, requiring access to a single master-keyed lock and an associated key.

For wireless, electronic access control systems, obtaining the information for copying a transponder can be even more straightforward. The necessary data could be, e.g., eavesdropped while they are exchanged via the RF interface. To counteract attacks, numerous manufacturers developed and offer (often proprietary) cryptographic solutions. Most of these have in common that they turned out to be insecure from a cryptanalytical perspective once the secret protocols and algorithms had been reverse-engineered. Today, most security products used for access control, e.g., Texas Instrument's Digital Signature Transponder (DST) [BGS+05], NXP's Mifare Classic cards [GKGM+08, Cou09], Hitag 2 transponders [VGB12], Legic Prime [PN09], and HID iClass (Elite) [GKGVM12] are considered broken: Mathematical attacks emerging from cryptographic weaknesses enable to circumvent the protection mechanisms in a limited amount of time (from a few minutes to several hours).

Sophisticated mathematical attacks on the SimonsVoss system 3060 exploiting weaknesses of the proprietary obscurity function (see Sect. 8.3) and a security vulnerability of the authentication protocol are presented in [SDK+13]. As a consequence of this traditional cryptanalysis, transponders with a known identifier can be duplicated in seconds. The attack exploits that an internal protocol value is re-used as a "random" number in the next protocol run and thus can be obtained by an adversary. However, this flaw is currently being fixed with a firmware update of the door lock, rendering the most severe mathematical attacks not applicable anymore.

8.1.2 Contribution

In contrast to the mathematical analyses, we illustrate how to circumvent the security mechanisms of SimonsVoss's system 3060 "G2" by means of physical attacks. In Sect. 8.2, we describe our attempts to recover the internals of the access control system and demonstrate how—despite the admittedly very time-consuming and demanding black-box analysis—an adversary can succeed in extracting SimonsVoss's undisclosed cryptographic protocols, cf. Sect. 8.3. As the main contribution, in Sect. 8.4 we present a side-channel attack targeting the key derivation function running on a lock. Our non-invasive SCA is able to extract the system key with approximately 150 EM measurements and enables access to all doors of an entire SimonsVoss installation. We finally discuss the requirements for SCA in the presence of obscurity countermeasures.

8.2 Reverse-Engineering a Black-Box System

Using the public information provided by SimonsVoss, it is impossible to determine the level of security for the system. Thus, in the following, we present the process of reverse-engineering the inner workings of the digital locking cylinder and the corresponding transponder.

8.2.1 Radio Protocol

The documentation available on the Internet [Sim13] yields only little information on the RF interface used to open a door. The transponder communicates with the counterpart in the door at 25 kHz at a specified maximum range of approx. 0.5 m. To capture the messages exchanged during an authentication, we used a simple coil made from copper wire and connected it to the USRP2, cf. Sect. 3.1.1. We eavesdropped on several (successful) protocol runs between a door and a transponder.

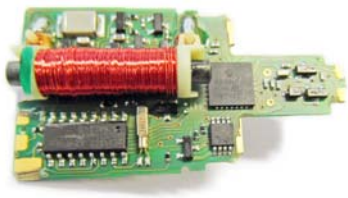
From studying and comparing various recorded messages, we understood the basic transmission principle and the encoding of the exchanged data. After further analysis of the interchanged messages, we found that some appear to be (pseudo-)random in each protocol run and thus that a cryptographic authentication scheme might be implemented.

8.2.2 Hardware and Circuit Boards

In order to understand the interaction between the mechanical and the electrical parts, we disassembled a door lock in a destructive manner, cf. Fig. 8.1a. We identified a magnetic pin that mechanically interlocks the bolt when voltage is applied to its contacts. The circuitry in the door thus controls whether the otherwise free-wheeling door knob is mechanically connected to the door latch and the deadbolt in order to grant access. No additional electronic circuits were found inside the lock, thus, all security functionality must be implemented on the door's PCB.

The PCB depicted in Fig. 8.1b is located inside one of the door knobs of the lock, which can be easily opened with a commercially available tool. The IC in the bottom-left corner is a proprietary SimonsVoss ASIC. On the right, a Microchip PIC16F886 µC [Mic09] is located. One connection between the `MCLR` pin of the PIC and the ASIC enables the latter to wake up the µC from the power save mode. The ASIC is also connected to the µC's `OSC1/CLKIN`, hence the µC can be clocked from the ASIC. Moreover, an external EEPROM in an SOT-8 package

(a) Door Hardware



(b) Lock

(c) Transponder

Figure 8.1: Mechanical parts of a door lock 3061 and the PCBs contained in the door knob and a transponder 3064

is connected to the µC. The plastic case of the transponder contains the PCB depicted in Fig. 8.1c. A comparison with the lock shows many identical parts, e.g., a PIC16F886 connected to an external EEPROM and the ASIC. A push-button triggering an authentication to the door is connected to the PIC.

The ASIC is a 16-pin IC with a custom label ("`MA124D`") for which we could find no information on the Internet. The pinout and the activity on the data lines did not point to any distinct manufacturer, thus, we decapsulated the silicon die of several ICs, cf. [Bec88, p.10]. Figure 8.2 depicts a high-resolution image of the silicon die. After analyzing microscopic pictures of the chip, it turned out that it contains a mask-configurable gate array IC with a rather limited amount of implementable logic. Besides, the ASIC does not feature internal memories.

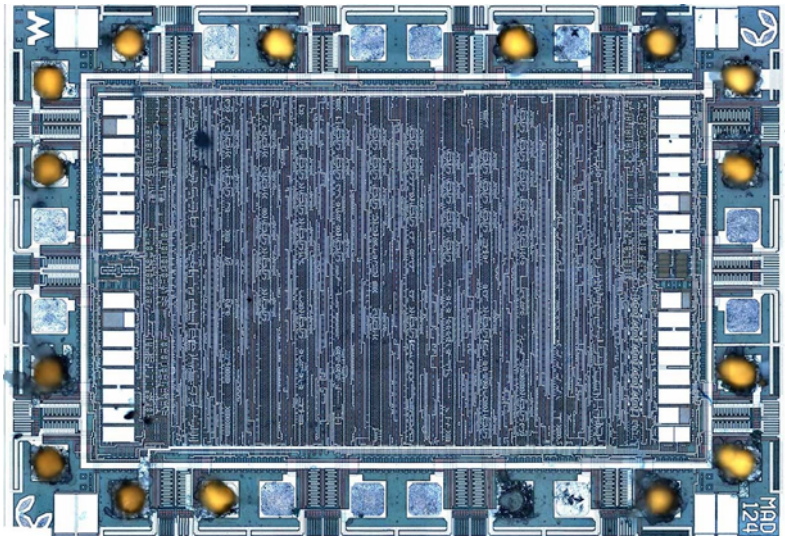


Figure 8.2: Microscope photograph of the SimonsVoss ASIC

We figured out that the majority of the available logic cells is used *(i)* for a counter that periodically wakes up the µC and *(ii)* for functions of the RF interface. It became clear that the ASIC does not contain security-related functions and thus can be ruled out as a target for further analyses. All relevant algorithms must thus be executed on the PIC µC. After analyzing

the radio protocol, the circuit boards, and the proprietary ASIC, we were unable to identify the security features of the SimonsVoss system 3060. To solve this problem, we decided to obtain the machine code of the PIC.

8.2.3 Extracting and Reverse-Engineering the Firmware of the PIC

After connecting a PIC programmer to the μC and trying to read out its content, it turned out that the read-out protection was enabled. Thus, following the methods proposed in [Zon11] and [Hua05], we tried to erase the code and data read-out protection bits: The PIC was decapsulated and the memory cells containing the protection bits exposed to Ultraviolet-C (UV-C) light. Even though Microchip covers the top of the respective cells of the PIC16F886 with small metal plates as a countermeasure, applying the UV-C light at an angle (so that it bounces off structures around the floating gate) deactivated the read-out protection. The whole process required less than 30 minutes. After that, the complete content of the PIC's program memory and its internal EEPROM could be extracted. We performed the read-out process for the PICs of several transponders and door locks and started to analyze their program code.

In order to disassemble and understand the extracted program code, we utilized the reverse-engineering tool IDA Pro [IDA]. Analyzing the program code, we were able to recover most previously unknown details of the SimonsVoss system 3060, including the authentication protocol and the employed cryptographic primitives, cf. Sect. 8.3. In addition to performing a static analysis of the program code, we also inserted a debug routine into the assembly code and re-programmed the PIC with our modified firmware.

With the capability to dump the memory contents, e.g., during the execution of a successful authentication protocol run, we were able to verify the results of the static analysis and to understand also those parts of the code that heavily depend on external input (e.g., from received data or data stored in the external memory). Obtaining and analyzing the program code was the essential step to enable a detailed examination of the system, including mathematical analysis and SCA. Being now able to understand the exchanged messages and the used cryptographic functions, the authentication mechanism described in Sect. 8.3 can be attacked in several ways, including the SCA presented in Sect. 8.4. Without the complete reverse-engineering, the SCA of the—in this regard extremely vulnerable—PIC μC would have been impossible.

8.3 SimonsVoss's Proprietary Cryptography

In this section, as a mandatory prerequisite for an SCA, we summarize the relevant results of reverse-engineering the code running on the PIC of the transponder and the lock. To this end, we describe the key derivation mechanism, the cryptographic primitives, and the protocol used for mutual authentication disclosed in [SDK+13]. The authentication protocol consists of in total eleven steps given in Fig. 8.3.

In the symmetric-key scheme, the transponder and the lock prove that they know a shared long-term secret K_T. On each transponder, this individual 128-bit key K_T is computed as the XOR of a 128-bit value $K_{T,\text{int}}$ stored in the internal EEPROM of the μC (not accessible from the outside) and a 128-bit value $K_{T,\text{ext}}$ stored in the (unprotected) external EEPROM, i.e., $K_T = K_{T,\text{ext}} \oplus K_{T,\text{int}}$. Each door within an entire SimonsVoss installation has an identical set of four 128-bit keys $K_{L,1}$, $K_{L,2}$, $K_{L,3}$, $K_{L,4}$, here called *system key*. Again, these keys are computed

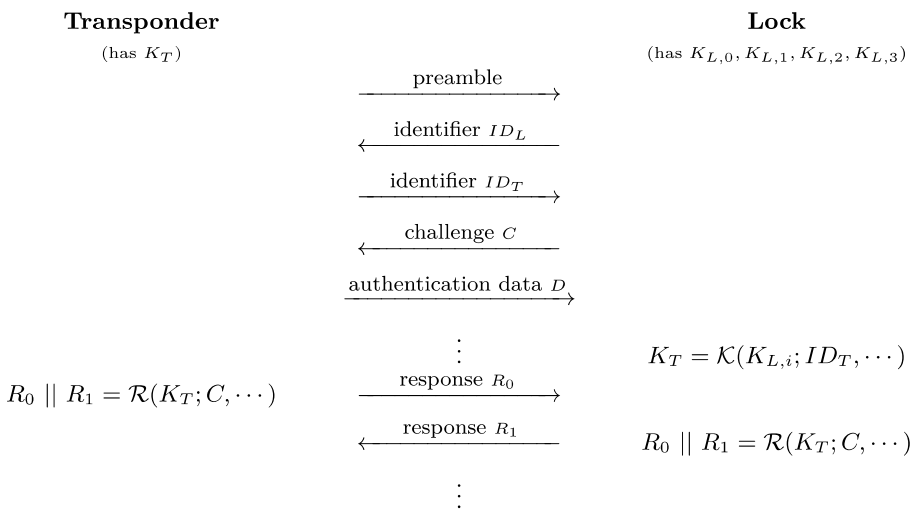


Figure 8.3: Protocol for the mutual authentication between a transponder and a lock

as the XOR of values contained in the internal and in the external EEPROM. However, the internally stored value is identical for all four keys, i.e., $K_{L,j} = K_{L,j,\text{ext}} \oplus K_{L,\text{int}}$. Note that when one of the four $K_{L,j}$ has been recovered, the remaining three can be determined after reading the respective values from the external EEPROM.

The system key is used to derive the key K_T of a transponder to be authenticated. After receiving the identifier ID_T of a transponder, the lock computes K_T on-the-fly using a key derivation function $\mathcal{K}$ involving the system key, previously exchanged authentication data D, and ID_T. All valid transponder identifiers that can authenticate to a particular door lock are stored unencrypted in its external EEPROM. Note that the key derivation is always executed by the door lock, even if it does not "know" the received ID_T.

The key derivation $\mathcal{K}$ consists of two building blocks: a modified DES denoted as $\mathcal{D}$ and a SimonsVoss-proprietary function $\mathcal{O}$ which we refer to as "obscurity function". $\mathcal{D}(K; M)$ is identical to a standard DES encrypting a 64-bit plaintext M under a 64-bit key K, except for the fact that some (security-unrelated) parts of the DES have been changed. We do not disclose the exact nature of this modification to protect the vendor.

$\mathcal{O}(Y; X)$ consists of eight rounds r, with $1 \leq r \leq 8$. It takes a 16-byte plaintext $X = ({x_0}^{(1)} \ldots {x_{15}}^{(1)})$ and a 16-byte key $Y = (y_0 \ldots y_{15})$ to compute a 16-byte ciphertext ${x_0}^{(9)} \ldots {x_{15}}^{(9)}$. Figure 8.4 depicts the structure of one round of $\mathcal{O}$. The internal "chaining values" ${c_1}^{(\mathrm{r})} \ldots {c_{15}}^{(\mathrm{r})}$ are processed horizontally in each round r and the last chaining value is used as the input ${c_0}^{(\mathrm{r}+1)}$ for the subsequent round $r+1$. The round constants $RC^{(\mathrm{r})}$ are fixed byte values, i.e., they are identical for every execution of $\mathcal{O}$, and added in each round after the first eight bytes have been processed. We do not give the values of theses constants here as part of a responsible disclosure process. The first chaining value is initialized with ${c_0}^{(1)} = RC^{(0)}$. All additions and shifts are performed modulo 256.

$\mathcal{K}$ takes the system key and a 128-bit parameter P_0 as inputs, where P_0 is derived from the first three bytes of ID_T and the first three bytes of the authentication data D as $P_0 =$

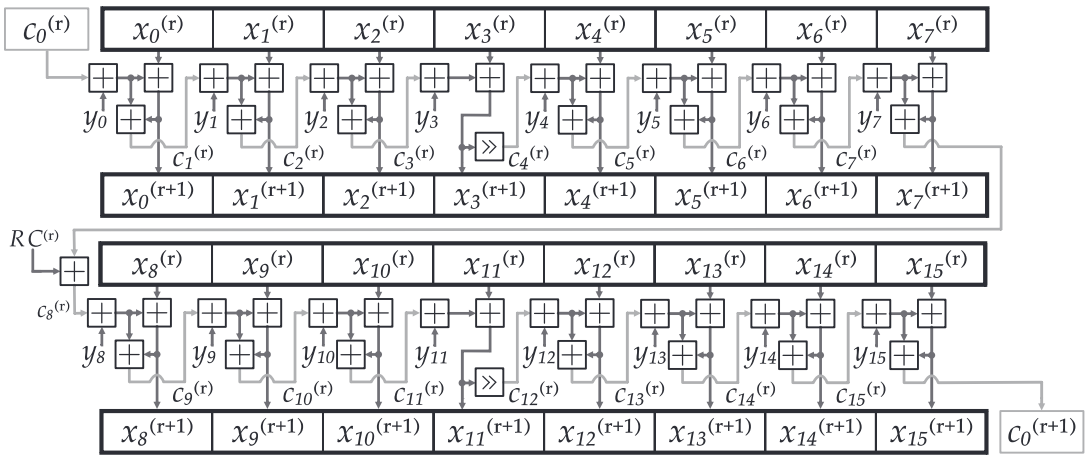


Figure 8.4: One round r of the obscurity function $\mathcal{O}$: The key bytes y_i are constant, while ${x_i}^{(\mathrm{r})}$ and the chaining values ${c_i}^{(\mathrm{r})}$ are updated in each round. The first chaining value is ${c_0}^{(1)} = RC^{(0)}$. The two 8-byte halves $x_0 \dots x_7$ and $x_8 \dots x_{15}$ are processed in an identical manner.

$(I_{T,0}, I_{T,1}, I_{T,2} \,\&\, \texttt{0xC7}, D_0, D_1, D_2 \,\&\, \texttt{0x3F}, 0, \dots, 0)$. The transponder key K_T is then computed by executing $\mathcal{O}$, $\mathcal{D}$, and $\mathcal{O}$:

$$\mathcal{K}(K_{L,j};\, P_0) = \mathcal{O}\left(P_0; \mathcal{D}\left(\mathcal{O}\left(K_{L,j}; P_0\right)_{64..127}; \mathcal{O}\left(K_{L,j}; P_0\right)_{0..63}\right) || 0 \dots 0\right).$$

Depending on the two Most Significant Bits (MSBs) of $I_{T,2}$, one of the four $K_{L,j}$ is selected as the key for the first (innermost) instance of $\mathcal{O}$ in $\mathcal{K}$ to encrypt P_0. The result is split into two 64-bit halves, the lower 64 bit being the plaintext and the upper 64 bit being the key of $\mathcal{D}$. The 64-bit result of $\mathcal{D}$ is then padded with 64 zero bits and encrypted with $\mathcal{O}$, this time using P_0 as the key. The resulting ciphertext is the transponder key K_T used in the subsequent steps of the challenge-response protocol. A lock is opened if K_T on the transponder and K_T derived by the door match.

To this end, both transponder and door compute a 64-bit response involving the challenge C and several protocol values as

$$\mathcal{R}\left(K_T; P_1, P_2\right) = \mathcal{D}\left(\mathcal{O}\left(\mathcal{O}(K_T; P_1); P_2\right)_{64..127}; \mathcal{O}\left(\mathcal{O}\left(K_T; P_1\right); P_2\right)_{0..63}\right),$$

with $P_1 = (C_0, \dots, C_{10}, D_6, \dots, D_9, 0)$ and $P_2 = (I_{L,2}, I_{T,2}, I_{T,3}, D_3, D_4, D_5, 0, \dots)$. The transponder sends the first 32-bit half R_0 of the output of $\mathcal{R}$, to which the door responds with the second half R_1 if R_0 was correct.

Obviously, the key derivation function $\mathcal{K}$ is the main target for an SCA, because recovering the system key allows to derive the key of any transponder in an installation given its ID_T.

8.4 Extraction of the System Key with SCA

In this section, we describe the steps to perform a side-channel attack on the SimonsVoss door lock 3061. As the main result, we show that the system key can be extracted from the employed

PIC µC with a non-invasive, CPA-based attack using approximately 150 traces. Possessing the system key, an adversary is able to create functionally identical clones of *all* transponders in an entire SimonsVoss installation. Given that we found numerous instances where the door lock was installed with the PCB being accessible from the outside, the attack poses a high practical threat in the real world. Note that we verified that the SCA can also be applied to a transponder in order to extract its key and duplicate it, but in a practical setting, it is highly unlikely that an adversary will take the efforts for the attack just for cloning a single transponder.

8.4.1 Theoretical Attack: Predicting Intermediate Values of the Obscurity Function in the Key Derivation

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

Note that only the first six bytes of the input to $\mathcal{O}$ can be chosen by the adversary because the remaining bytes of P_0 are set to zero. Due to the properties of $\mathcal{O}$, these input bytes do not lead to a "full" randomization of all intermediates in the first round of the obscurity function. A straightforward CPA of the first round targeting the addition operation ${x_i}^{(\mathrm{r}+1)} = {x_i}^{(\mathrm{r})} + {c_i}^{(\mathrm{r})} + y_i$ hence only allows to recover the first seven key bytes $y_0 \ldots y_6$.

For the remaining key bytes, at least a part of the addition in the first round is carried out with completely constant data, ruling out a CPA: For revealing y_7, one obtains already two candidates, because the LSB of ${c_7}^{(1)}$ does not depend on the varying part of the input and hence remains constant. For ${c_8}^{(1)}$, two bits are not randomized, leading to four candidates for y_8 (if c_7 was known).

Thus, to obtain these two bytes, two CPAs each with four additional candidates would be necessary. To recover all key bytes using this approach, an overall number of $2^{1+2+3+4+5+4+5+6+7} = 2^{37}$ key candidates would have to be tested, leading to an impractical attack. Hence, we utilize further properties of $\mathcal{O}$ to reduce the computational cost by extending the attack to initially non-recoverable key bytes in subsequent rounds of $\mathcal{O}$.

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

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

In the third, fourth, and fifth round, only the MSBs of ${c_0}^{(3,\,4,\,5)}$ vary. The remaining bits are constant and can be recovered. In addition, due to the multiplication by two (i.e., a binary

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

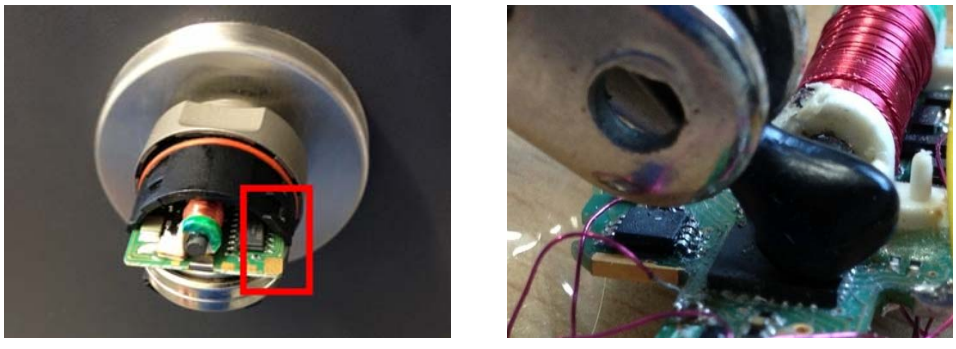
In short, the attack can be summarized as follows: First, one recovers the first seven key bytes in round one. Then, the constant part of c_0 at the beginning of each round has to be found. Finally, all key bytes for which the update operation is fully randomized in the respective round can be obtained.

8.4.2 Practical Results

In the following, we first describe the measurement setup used to acquire side-channel traces for the SimonsVoss lock 3061. Using techniques and results of the reverse-engineering described in Sect. 8.2, we profile the DUT and determine the point in time where the leakage occurs in a trace. Finally, applying the attack described in Sect. 8.4.1, we recover the system key using a limited number of traces in a non-invasive manner.

Measurement and Profiling

To trigger the key derivation on the DUT, we directly connected to the data lines between the ASIC and the PIC and controlled the relevant control signals using the GIAnT (Chap. 5). The respective pins (marked with a red rectangle in Fig. 8.5a) are accessible without removing the PCB from the door. Equivalently, the communication with the DUT could also be performed sending data over the RF interface, e.g., using the USRP2 as described in Sect. 8.2.1. We primarily decided not to use the RF interface to increase the (mechanical) stability of the measurement setup.



(a) PCB in the door part (b) EM probe on PIC

Figure 8.5: Setup for SCA of the door part

We non-invasively measured the power consumption during the execution of the key derivation function for randomly chosen ID_T and authentication data D (apart from a few bits that are required to be set by the protocol). To this end, we placed an EM probe (cf. Sect. 3.2.2) on the package close to the power supply pins of the PIC as depicted in Fig. 8.5b. Initial experiments to measure the power consumption by inserting a shunt resistor into the battery connection yielded heavily smoothed power traces, presumably due to several bypass/voltage stabilization capacitors on the PCB. Moreover, the ASIC is also connected to the main battery supply, resulting in wide-band, high-amplitude noise that could not be filtered out.

In contrast, the EM probe at the given position mainly picked up the power consumption of the PIC. Using the described setup, we recorded 1,000 traces using the Picoscope 5204 (Sect. 3.2.1) at a sample rate of 500 MHz. With the current measurement setup, due to delays caused by the protocol, approximately 10 traces can be acquired per minute. Note that the PIC runs on the internal RC oscillator at a frequency of approximately 8 MHz. Hence, we further (digitally) bandpass-filtered the recorded traces. Experimentally varying the lower and upper frequency of the passband, we found that a passband from 6 MHz to 9 MHz yields the best results.

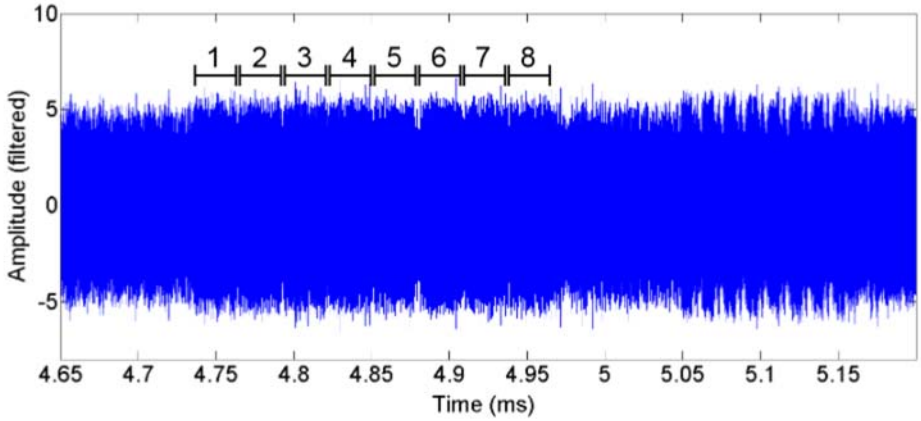
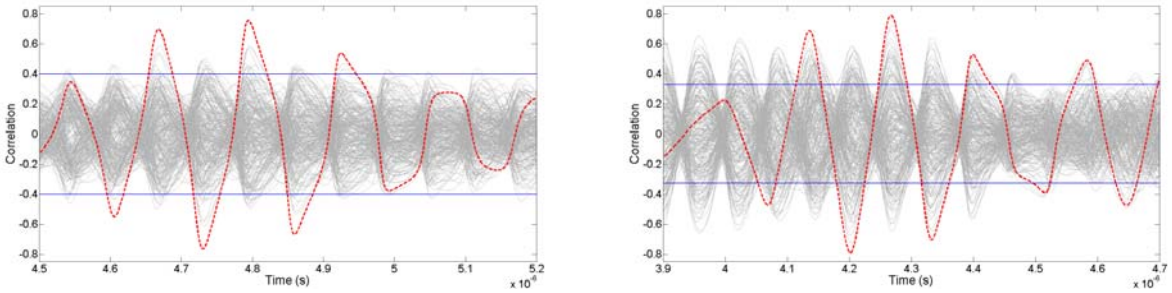


Figure 8.6: Filtered EM trace during the execution of the first obscurity function $\mathcal{O}$ in the key derivation, eight rounds visible as eight marked patterns

To determine the relevant part of the trace belonging to the key derivation, we initially inserted a small function into the (otherwise unmodified) code of the PIC. This function generated a rising edge on an unused pin of the PIC, serving as a trigger signal for profiling purposes. Figure 8.6 exemplarily depicts the part of a trace belonging to the initial execution of $\mathcal{O}$ in the key derivation. The eight rounds of the function can be recognized as eight distinct "humps". Furthermore, a unique pattern occurs at the beginning of the relevant part (and each round). This pattern serves for alignment purposes: Having profiled the DUT, we removed the artificial trigger signal and recorded traces for the original, unmodified code. We then used the pattern to align the traces for the actual CPA attack.

Attack Results

Having determined the relevant part of the trace and the appropriate pre-processing steps, we practically performed the (theoretical) attack described in Sect. 8.4.1. We use the HD between a byte ${x_i}^{(\mathrm{r})}$ and its updated value ${x_i}^{(\mathrm{r}+1)}$ as the power model. This model was derived based on the analysis of the code implementing $\mathcal{O}$ and the leakage model for the PIC series described in [Gol08]. Using this model, we obtained correlation values of approximately 0.75 for the correct candidate, while all other (wrong) candidates exhibited a lower value. This allows to unambiguously determine the key with approximately 100 traces, as exemplarily depicted in Fig. 8.7a.



(a) y_7 in Round 2

(b) ${c_0}^{(2)}$ in Round 2

Figure 8.7: Correlation for key byte y_7 after 100 traces and ${c_0}^{(2)}$ after 150 traces. Correct candidate: red, dashed

The CPAs for ${c_0}^{(\mathrm{r})}$ exhibit a similar behavior, cf. for instance Fig. 8.7b for ${c_0}^{(2)}$. However, for determining the partial ${c_0}^{(\mathrm{r})}$ in later rounds, a (slightly) higher number of traces is needed (especially for round six and seven), since only the LSBs of the respective intermediate values are predicted. Still, approximately 150 traces were sufficient to obtain the maximum correlation at the correct point in time for the correct candidate.

Note that—at a higher computational cost—the CPA on ${c_0}^{(6)}$ and ${c_0}^{(7)}$ could be left out altogether: The attack on the actual key bytes in round six and seven could be carried out for all 2^5 or 2^3 candidates for ${c_0}^{(6)}$ or ${c_0}^{(7)}$, respectively. This would increase the number of candidates to $2^5 \cdot 2^8 = 2^{13}$ and $2^3 \cdot 2^8 = 2^{11}$. For this amount of candidates, a CPA can still be executed efficiently.

8.5 Conclusion

In this chapter, we presented a practical, non-invasive side-channel attack on the SimonsVoss system 3060 “G2”, allowing to extract the system key from one door lock in an installation. The attack requires approximately 150 EM traces, which can be recorded in approximately 15 minutes using our current measurement setup. An adversary possessing the system key ultimately gains access to all doors of the entire installation.

Reverse-engineering the embedded code of the PIC µC enabled the profiling and the development of a suitable prediction function for the CPA. Surprisingly, the cryptanalytical weakness of the employed obscurity function (highly linear structure, slow avalanche effect) together with

its specific usage in the key derivation (input largely constant) required a more complicated SCA compared to attacking a standard cipher, e.g., DES or AES.

Preventing the SCA without replacing hardware components is very difficult in our opinion. The program memory and the RAM of the PIC of the lock are almost fully utilized. This rules out the implementation of SCA countermeasures, e.g., masking, requiring an (even slightly) increased program size or RAM usage. Randomizing the timing of the algorithms appears likely to be ineffective, because *(i)* clear patterns in the EM trace can be detected and *(ii)* no suitable entropy source is available. Furthermore, all countermeasures could again be reverse-engineered and closely inspected on a program-code level.

The most important lesson to be learned from the demonstrated attack is that the overall security of a system depends on the weakest link in the chain: The cryptographic algorithm used by SimonsVoss, i.e., a DES core combined with an obscurity function that obfuscates the input and output bytes of the DES and enlarges the key length to 128 bit, seemed to provide a reasonable level of security. As long as the proprietary scheme remained undisclosed, neither a brute-force attack nor mathematical cryptanalysis of the unknown cipher were a practical option. Likewise, an analysis of the protocol and SCA attacks targeting a hypothesized cipher were fruitless.

One single factor, however, compromised the security of the SimonsVoss system: a vulnerability of the PIC μC that enables to bypass the code read-out protection. Without this weakest link of the chain and the respective central step of the analysis—reverse-engineering the program code—probably no (mathematical and implementation) attacks would have been found.

Chapter 9

The Yubikey One-Time Password Token

The classical way of authentication with a username-password pair is often insufficient. An adversary can choose from a multitude of methods to obtain the credentials, e.g., by guessing passwords using a dictionary, by eavesdropping on network traffic, or by installing malware on the system of the target user. To overcome this problem, numerous solutions incorporating a second factor in the authentication process have been proposed. A common approach is to provide each user with a hardware token that generates a One-Time Password (OTP) in addition to the traditional credentials. The token itself comprises a secret cryptographic key that, together with timestamps and counters, is used to derive a fresh OTP for each authentication. A relatively new yet wide-spread example for an OTP token is the Yubikey 2 produced by Yubico. This device employs an open-source protocol based on the mathematically secure AES and emulates a USB keyboard to enter the OTP in a platform-independent manner. In this chapter, we analyze the susceptibility of the Yubikey 2 to side-channel attacks. We show that by non-invasively measuring the power consumption and the EM emanation of the device, an adversary is able to extract the full 128-bit AES key with approximately one hour of access to the Yubikey 2. The attack leaves no physical traces on the device and can be performed using low-cost equipment. In consequence, an adversary is able to generate valid OTPs, even after the Yubikey 2 has been returned to the owner.

Contents of this Chapter

9.1 Introduction

Considering the steadily increasing risk due to, e.g., phishing and malware, normal authentication schemes like username and password are not sufficient anymore for high-security (online)

services. Therefore, means to strengthen the authentication by introducing an additional factor are mandatory. A popular example of these techniques are OTPs generated by a hardware token. These tokens are common in high-security commercial applications, but not (yet) for private use, often because of their high price and the need for additional sever infrastructure.

Attacks on two-factor authentication systems, most prominently the breach of RSA's SecurID system [Bri11, BFK+12], were until now mostly based on weaknesses in the cryptographic design of the protocol or the backend network. In contrast, attacks on the actual (hardware) implementation are assumed to have much higher requirements with respect to the capabilities of an adversary. Indeed, "classical" invasive attacks on modern OTP devices use expensive equipment, e.g., microprobes or a Focused Ion Beam (FIB) that can only be operated by an experienced semiconductor engineer. Therefore, the question arises if OTP tokens are susceptible to more cost-effective attack methods like SCA. In this chapter, we use the example of the Yubikey 2, a USB-based device manufactured by Yubico Inc. [Yub12]. It differs from most other OTP tokens with a focus on simplicity and an open-source software backend. The question arises if high-security requirements can be fulfilled by such a low-cost device and how well the token protects the 128-bit AES key used for the OTP generation. Yubico has several security-sensitive reference customers (that use the Yubikey, e.g., for securing remote access) listed on their website [Yub13a], for example, Novartis, Agfa, and U.S. Department of Defense Contractors. The U.S. Department of Defense Contractors even switched from RSA's SecureID system to the Yubikey [Yub], even though the Yubikey 2 is not certified for governmental standards.

The work described in this chapter was jointly done with Bastian Richter. The results will be published at RAID 2013 [ORP13]. Acknowledgments go to Christoph Wegener for his remarks and contributions in the course of our analysis.

9.1.1 Two-Factor Authentication

As mentioned above, the "normal" way of authentication by means of username and password is not sufficient in many cases. The credentials can often be obtained, e.g., by social engineering or due to protocol weaknesses (cf. [PA13] for a recent example). Thus, an additional security factor is needed. An established solution for this problem are OTPs. An OTP is generated by a hardware (or sometimes software) token and provided in addition to the normal credentials. The token generates a value which is valid for a single use, sometimes also only for a short period of time. Now, the user has to *know* the username and password and additionally has to *own* the token to successfully perform an authentication. The OTP is usually derived based on usage counters, timestamps, and a secret key securely stored on the token, by, e.g., hashing or encrypting the respective values.

Of course, if an adversary manages to obtain both the physical token and the credentials of the user, he is able to gain unauthorized access. However, as soon as the token is returned to the owner in order to conceal the attack, the adversary is no longer able to impersonate the rightful user.

9.1.2 Adversary Model

We assume an adversary gaining physical access to the token for a limited amount of time (in the range of a few hours), e.g., when a user left his token at his desk. Besides, a token could also be stolen and returned without the owner noticing. Especially in the light of, for

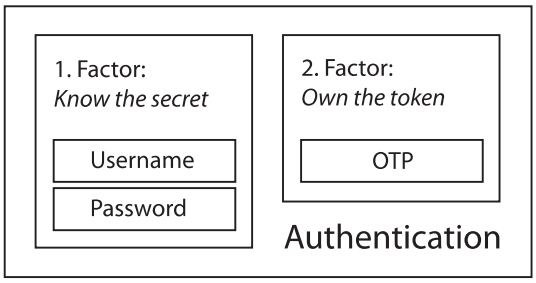


Figure 9.1: Principle of authentication with two factors

example, the attack on Lockheed Martin presumably being the motivation for the intrusion into RSA's network [Bri11], this scenario is less hypothetical than it initially sounds. Organizations specialized in industrial espionage go to great lengths to overcome protection mechanisms, and obtaining a user's token for a limited amount of time seems conceivable.

In contrast to just using the token to login and then returning it, we focus on an attack that actually extracts the cryptographic secret from the device. This allows an adversary to create indistinguishable clones of the original device, usable for an unlimited amount of time. Apart from having direct access to the device, no modifications or invasive steps, e.g., the decapsulation of the token, are required. The SCA described in this section is based on the non-invasive, passive observation of the token's behavior and hence does not leave physical traces that can be detected later.

9.1.3 Related Work

The security of—today heavily outdated—USB tokens was analyzed in [Gra00], describing hardware and software weaknesses but not covering side-channel attacks. In [Gra04], it is stated that newer devices are harder to attack and that a "lunchtime attack [is] likely not possible". For the SecurID tokens manufactured by RSA, there are reports on both attacks on the backend [Bri11] and flaws on the protocol level [BFK+12]. However, the real-world relevance of the latter attack is denied by RSA [Cur12].

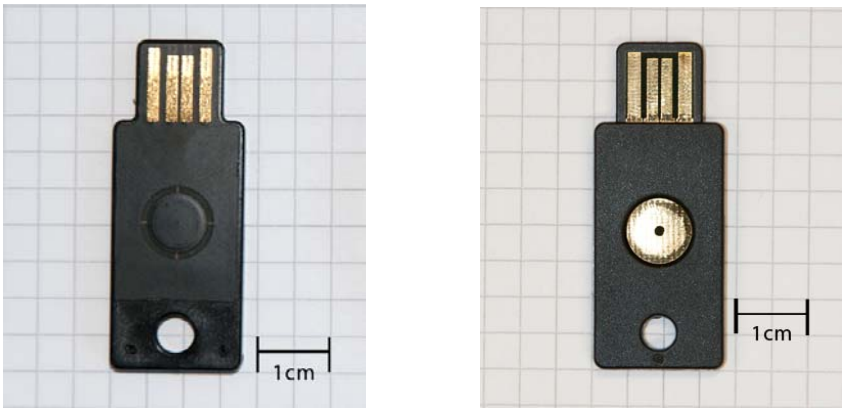
The cryptanalytical security of parts of the protocol used for the Yubikey was analyzed in [Vam12], and no severe formal vulnerabilities were found. Yubico mentions the threat of side-channel attacks in a security evaluation on their website [Yub13e], however, apparently did not further investigate this issue.

9.1.4 Contribution

The remainder of this chapter is organized as follows: in Sect. 9.2, we describe the OTP generation scheme and analyze the underlying hardware of the Yubikey 2. The measurement setup for acquiring power consumption and EM traces for an SCA is presented in Sect. 9.3. In Sect. 9.4, we detail on the initial side-channel profiling of the Yubikey 2, leading to the full key-recovery attack shown in Sect. 9.5. We conclude in Sect. 9.6, discussing suitable countermeasures and describing the reaction of the vendor Yubico, which we informed ahead of time as part of a responsible disclosure process.

9.2 The Yubikey 2

We analyzed the current version 2 of the Yubikey Standard with the firmware version 2.2.3. The predecessor Yubikey 1 (cf. Fig. 9.2a) was introduced in 2008 but already replaced by the current Yubikey 2 (Fig. 9.2b) in 2009 [Yub12]. Apart from the Yubico-specific OTP generation, the Yubikey 2 can also be used to store a static password. Besides, the Yubikey 2 can be used as a token for generating HMAC-based One Time Password (HOTP) specified by the Initiative of Open Authentication (OATH) [Yub13a]. However, we do not further examine these additional features in this chapter and focus on the default OTP mechanism.



(a) Yubikey 1 (b) Yubikey 2

Figure 9.2: The two versions of the Yubikey Standard

9.2.1 Typical Use

The Yubikey 2 appears as a normal USB keyboard to the user's computer to enable direct input of the OTP. An advantage of this technique is that it does not require an extra driver installation and works with default keyboard drivers available on virtually every relevant operating system. When the user presses the button on top of the DUT, the token generates a OTP, encodes it in a specific format described in Sect. 9.2.2, and enters it using simulated keyboard inputs. The intended way of using the OTP is depicted in Fig. 9.3. The user first enters his credentials and then gives focus to an additional input field on the login form before pressing the Yubikey's button.

9.2.2 OTP Structure

The OTP generated by the Yubikey 2 is based on several counters, random bytes, a secret ID, and a checksum, which are concatenated to a 16-byte value and subsequently encrypted using the AES with a 128-bit key.

UID The private ID is 6 byte long and kept secret. It can be used as another secret parameter or to distinguish users when a common encryption key is used.

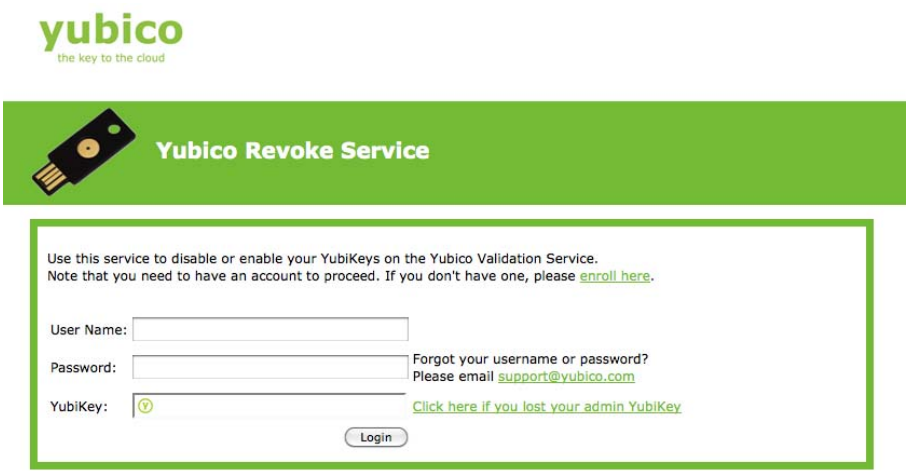


Figure 9.3: Typical Yubikey login form

useCtr The usage counter is 2 byte long and increased at the first OTP generation after power-up. Additionally, the counter is incremented when the session usage counter wraps from `0xff` to `0x00`.

tstp The 3-byte timestamp is initialized with random data on startup. After this, the timestamp is incremented with a frequency of approximately 8 Hz.

sessionCtr The 1-byte session counter starts with `0x00` at power-up and is incremented on every OTP generation.

rnd 2 additional byte of random data.

crc A 2-byte CRC16 checksum computed over all previous data fields.

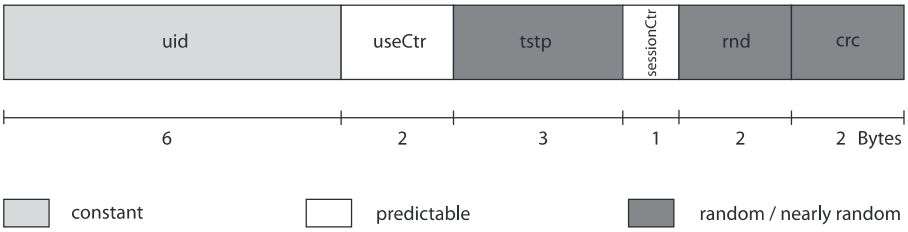


Figure 9.4: Structure of a Yubikey OTP

Figure 9.4 gives an overview of the structure of the OTPs and indicates which fields are static, predictable, or random.

All data fields are concatenated and then AES-encrypted using the secret 128-bit key programmed into the Yubikey 2. Usually, this key is set once by, e.g., the system administrator using the configuration utility [Yub13b] before the Yubikey 2 is handed to the user. The resulting ciphertext of the AES encryption is encoded using a special encoding called “Modhex”

to avoid problems with different keyboard layouts by limiting the simulated key presses to alphanumeric characters that have the same keycode in most locales. To identify the Yubikey, a Modhex-encoded 6-byte public ID is pre-pended to the encoded ciphertext.

To verify the OTP, the server-side software, e.g., the open-source validation server provided by Yubico, undoes the Modhex encoding, retrieves the AES key stored for the respective public ID, decrypts the OTP, and validates the resulting data. More precisely, the following steps are performed for the verification of an OTP:

(1) Identify the Yubikey by the public ID and retrieve the corresponding AES key

(2) Decrypt the OTP with the corresponding AES key

(3) Check the CRC checksum

(4) Check if the private ID is correct

(5) Compare the counters to the last saved state:
   - Accept the OTP if the session counter has been incremented or
   - If the session counter has not been incremented, accept the OTP if the usage counter has increased

If the OTP does not meet one of the previous conditions, it is considered invalid and the authentication fails. The conditions for the counters are in place to avoid replay attacks.

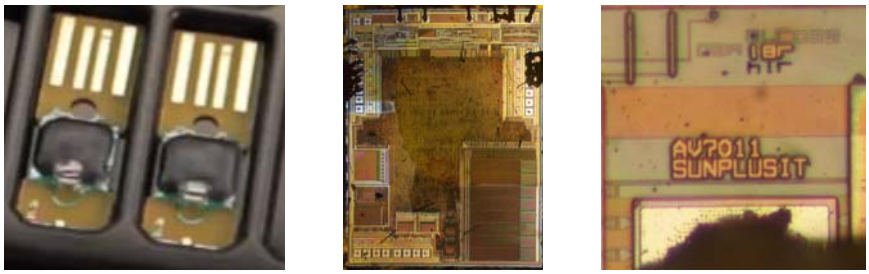
9.2.3 Hardware of the Yubikey 2

The Yubikey 2 is mono-block molded and thus hermetically sealed. To find out which kind of µC is used in the Yubikey 2, we dissolved the casing with fuming nitric acid to gain access to the silicon die (cf. Fig. 9.5a). The position of the µC was known from a promotional video about the production of the Yubikey [Yub13c], from which we extracted the picture of the PCB shown in Fig. 9.5a. On the die, we found the label "SUNPLUSIT" (cf. Fig. 9.5c) which seems to belong to Sunplus Innovation Technology Inc. based in Taiwan [Sun]. We were unable to exactly find out which controller was used, as there is no Sunplus part related to the label "AV7011". However, all Human Interface Device (HID) µCs produced by Sunplus employ an 8-bit architecture. This fact is important when searching for a suitable power model for the SCA.

9.3 Measurement Setup

To record power traces for an SCA, we built a simple adapter to get access to the USB power and data lines, cf. Fig. 9.6a. Note that the developed measurement adapter is not specific to the Yubikey, but can be used in general for power measurements of USB devices. The basic setup gives simple access to the USB lines and provides a pin to insert a shunt resistor for power measurements. The `D+` and `D-` lines were directly connected to the PC's USB port. A 60 Ω resistor was inserted into the ground line to measure the power consumption of the Yubikey.

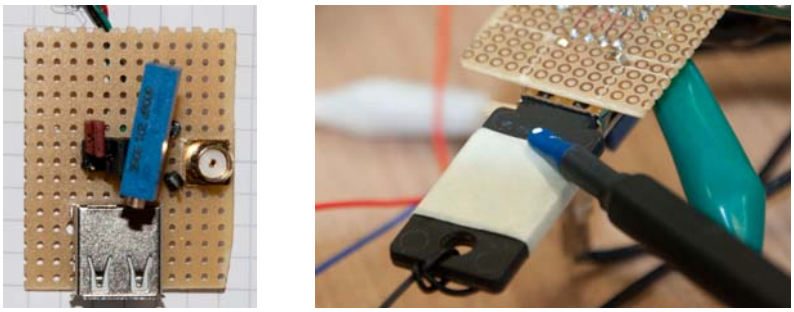
In our first experiments, we used `Vcc` provided by the USB port as power supply for the Yubikey, however, this resulted in a high amount of measurement noise. Therefore, an external



(a) PCB from [Yub13c] (b) Complete die (c) Die label

Figure 9.5: Die of the μC in the Yubikey 2

power supply was added to reduce the noise caused by the PC's power supply. Figure 9.7a depicts the overall structure of the measurement setup.

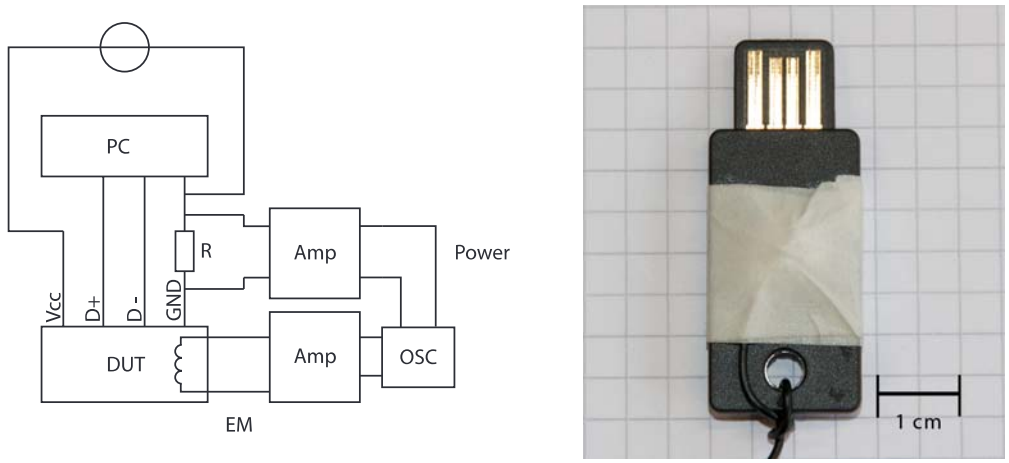


(a) USB adaptor (b) EM probe

Figure 9.6: Measurement methods: USB adapter with shunt resistor and EM probe at the position with maximal signal amplitude on the Yubikey 2

A custom amplifier was added to amplify the measured voltage drop over the resistor. This was necessary because the measured (unamplified) voltage was too low to fill the minimal input range of ±100 mV of the utilized Picoscope 5204 DSO (cf. Sect. 3.2.1). All measurements were recorded at a sample rate of 500 MHz. Initially, to perform the profiling of the DUT described in Sect. 9.4, we focused on the power consumption measured via the shunt resistor. However, in subsequent experiments and for improving the key-recovery described in Sect. 9.5, we also recorded the EM emanation of the DUT by placing a near-field probe on an experimentally determined position on the package of the Yubikey 2. The resulting signal was amplified by 30 dB using the Langer EMV amplifier, cf. Sect. 3.2.2. The EM probe on the casing to the Yubikey is depicted in Fig. 9.6b. The overall cost for the setup used in this chapter is approximately USD 3,000. Hence, the attack described in Sect. 9.5 can be performed at low cost without sophisticated, expensive lab equipment.

To initiate an encryption on the Yubikey, a capacitive button on top of the token has to be pressed. This button is basically a open plate-type capacitor whose capacitance changes when a finger is placed on top. For our purposes of automatic measurements, manually pressing the button is not an option. However, the finger can be "simulated" by connecting the blank metal



(a) Schematic

(b) Yubikey 2 with wire for simulating button presses

Figure 9.7: Setup for measuring the power consumption and EM emanation

contact on top of the Yubikey 2 to ground. For this purpose, we used a MOSFET transistor controlled by an ATXMega μC. The Yubikey with the controlling wire is depicted in Fig. 9.7b.

Note that this setup is not fully stable. This can lead to false button presses or failures to press the button at all. Thus, the measurement software was prepared to handle these problematic cases.

9.4 Side-Channel Profiling

The data acquisition of the DSO was triggered using a large drop within the power consumption of the device caused by the status LED of the Yubikey being turned off. A level dropout trigger—firing when the signal has been below a certain level for a defined period of time—was employed. Note that the DUT needs at least 2.6 seconds to "recover" after a button press. Incidentally, this significantly slows down the measurement process (and thus the overall attack) because the speed of the data acquisition is limited by this property of the Yubikey.

There are glitches regularly occurring in the power traces. These glitches are apparently generated by the DUT and do not occur when simply measuring the supply voltage without the DUT being connected. They follow a constant interval of 1 ms, but do not have a constant offset to the voltage drop. Because of this, they might be caused by the USB Start Of Frame (SOF) packets that are sent by the PC in an 1 ms interval actively polling the DUT. These glitches turned out to be problematic because they have a large influence on the amplitude of the trace and disturb the statistical methods used in the subsequent analysis. In order to solve this problem, a MATLAB function was developed to detect these wide glitches and discard the respective power trace. As a result, the effective number of power traces usable for SCA is approximately 65 % of the overall number of recorded traces.

9.4.1 Locating the AES Encryption

When initially examining the power trace of the DUT, the significant voltage drop caused by shutting off the LED was used as a reference point. Right before the voltage drop, a pattern can be observed that resembles a structure with ten rounds, each approximately 200 µs long, cf. Fig. 9.8.

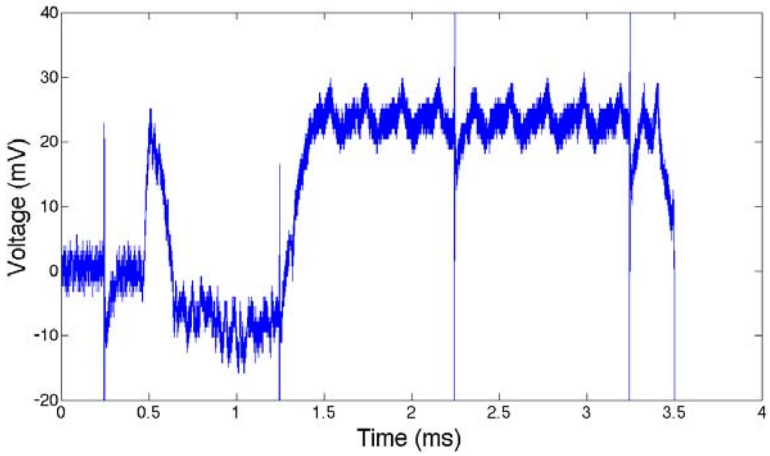


Figure 9.8: Ten-round pattern in the power traces before the LED is being shut off

Since the AES-128 employed on the Yubikey has ten rounds, it is likely that this part of the trace belongs to the AES encryption. This is further confirmed by Fig. 9.9 showing an average trace computed using 1000 power traces. The "rounds" are clearly visible, and even different operations are distinguishable within one round. Note that, however, we were unable to observe single instructions within one round, rather, it appears the traces are in some way lowpass filtered. This may, for instance, be due to decoupling capacitors or an integrated voltage regulator of the µC. Additionally, the tenth round at approximately 2.1 ms is 70 µs shorter than the others, which agrees with the fact that the final round of the AES algorithm misses the `MixColumns` step.

We recorded 20,000 traces of the part presumably belonging to the AES operation. The 128-bit AES key was set to `ad 5c 43 c5 2f 25 a7 4a 94 41 c2 1f 35 5b 43 09`. The used sample rate was 500 MHz as mentioned in Sect. 9.3. Experimentally, we found that (digitally) downsampling the traces by a factor of ten does not affect the success rate of the subsequent attack presented in Sect. 9.5. Hence, to reduce the data and computation complexity, all experiments described in the following included this pre-processing step.

We tested different models for the power consumption of the device. An 8-bit HW model for single bytes of the intermediate values within the AES turned out to be suitable, confirming the assumption that an 8-bit µC is used in the Yubikey 2. To identify the different AES operations within the rounds, a CPA using the HW of certain output bytes of the S-boxes in round nine and of certain input bytes to round ten as the power model was performed. The correlation results (after 6,400 power traces) can be exemplarily seen for byte 13 and 16 in Fig. 9.10. The horizontal blue lines at ± 0.05 indicate the expected noise level of $4/\sqrt{\#\,\text{traces}}$, cf. Sect. 2.3.2. A correlation exceeding this boundary is considered significant, i.e., means that the DUT performs

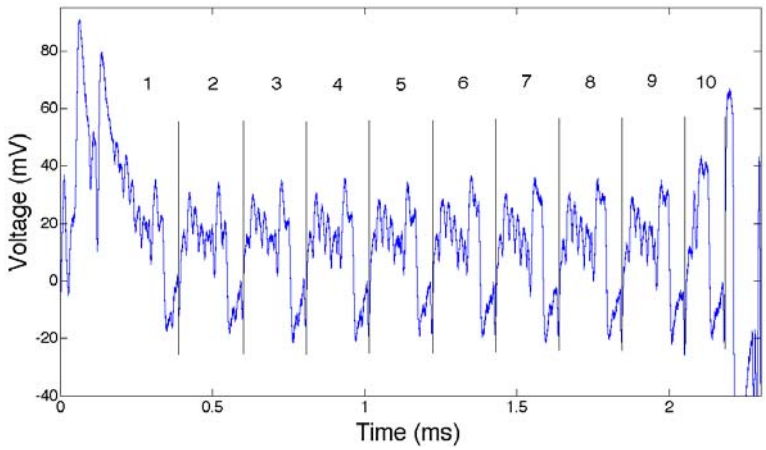


Figure 9.9: Average over 1,000 amplified traces of the part suspected to belong to the AES encryption

a computation involving the predicted value (in this case state bytes in round nine and ten) at the respective point in time.

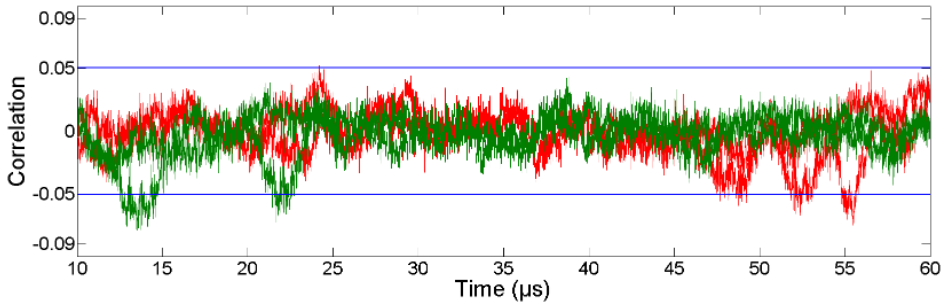


Figure 9.10: Correlation for byte 13 and 16, HW of the S-box output in round nine (green, 10...25 µs) and HW of the input to round ten (red, 50...60 µs) using 6,400 traces

9.4.2 EM Measurements

As mentioned in Sect. 9.3, we also captured the EM emanation of the DUT at the same time as the power consumption in subsequent experiments. The EM traces mainly showed a clock signal at a frequency of 12 MHz. However, digitally amplitude-demodulating [Sha06] this signal yielded a trace not exhibiting the lowpass filtered shape observed for the power traces. Figure 9.11 depicts a power trace (blue, bottom) and the corresponding demodulated EM trace (green, top). In both cases, the round structure is discernible. Yet, the EM trace allows to separately observe every clock cycle, while the power consumption trace only shows the overall round structure.

Similar to the power consumption traces, we also observed distorted EM traces. However, the overall number of "usable" traces was higher compared to the power consumption measure-

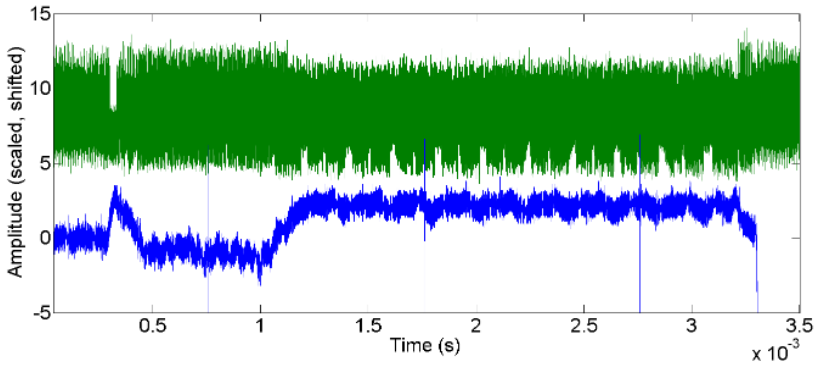


Figure 9.11: Power trace (blue, bottom) and demodulated EM trace (green, top). Vertical scaling and offset changed to compare general signal shape

ment: only 25 % of the EM traces had to be discarded, compared to about 35 % for the power consumption traces.

9.5 Extracting the AES Key

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

9.5.1 Key Recovery using Power Traces

We computed the correlation coefficient for all 256 candidates for each key bytes using 10,000 traces. The hypothetical power consumption h_i was predicted as $h_i = \mathrm{HW}\left(\mathrm{SBox}^{-1}\left(C_i \oplus rk\right)\right)$, with C_i a ciphertext byte (for measurement i) and rk the corresponding byte of the round key (dropping the byte index for better readability). The correlation coefficients for the first, second, eighth, and ninth key byte are exemplarily shown in Fig. 9.12. Evidently, the maximum absolute value for the correlation coefficient occurs for the correct key candidate. This observation also hold for the remaining bytes, for which the results are not depicted in Fig. 9.12.

To get an estimate of how many traces are needed to clearly distinguish the correct key candidate from the wrong ones, the maximum correlation coefficient (at the point of leakage) for each candidate after each trace was saved. The result after 10,000 traces is depicted in Fig. 9.13 for all key bytes.

We used the ratio between the maximum correlation for the correct key candidate and the highest correlation for the "second best" wrong candidate as a metric, cf. Sect. 2.5. We then used the number of traces for which this ratio is greater than 1.1 as the minimum number of required traces given in Tab. 9.1.

We were able to clearly determine the full 128-bit AES key using approximately 4,500 traces. In this regard, it turned out that the number of traces needed to recover a key byte differs: For byte 1, 4, and 16, less than 1,000 traces were sufficient. For byte 8, 9, 10, 11, 13, and 14, less

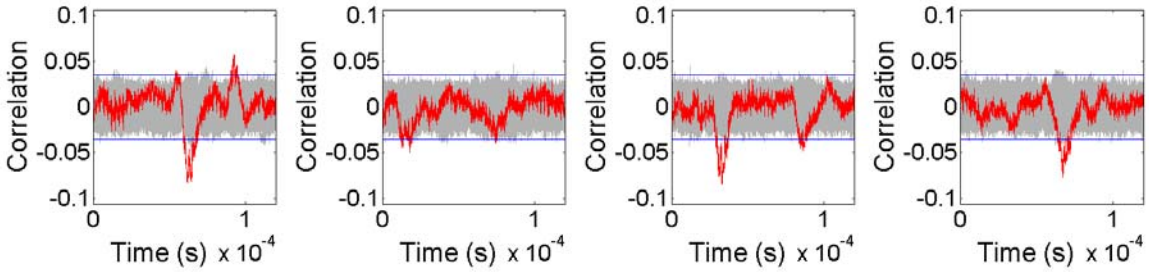


Figure 9.12: Correlation coefficient for all candidates for the key bytes 1, 2, 8, and 9 (left to right) after 10,000 power traces. Red: correct key candidate, gray: wrong key candidates

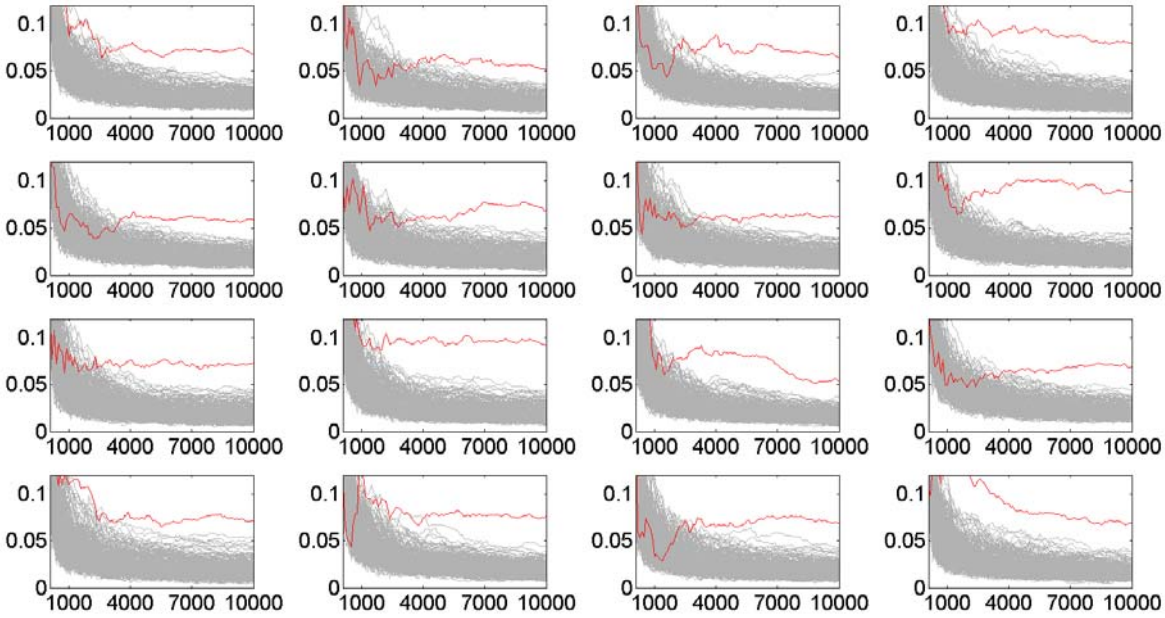


Figure 9.13: Evolution of the maximum correlation (vertical axis) for power measurements over the number of used traces (horizontal axis) for all key bytes (left to right, top to bottom). Red: Correct key candidate

than 3,000 traces sufficed to determine the correct value. For byte 2, 3, 5, 6, 7, 12, and 15, a number between 3,100 and 4,500 traces lead to the correct key byte being found.

Note that the pre-selection of the traces necessary due to the glitches mentioned in Sect. 9.4 effectively requires more traces to be recorded: for 4,500 usable traces, approximately 7,000 traces had to be measured in total. With our current measurement setup, 1,000 traces can be acquired in about 1.5 h, i.e., at a rate of 11.1 $^{\text{traces}}/_{\text{min}}$. Thus, to obtain 7,000 traces in total, approximately 10.5 h of access to the DUT are necessary.

The "spread" correlation peak with a width of 8.3 µs would translate to a clock frequency of approximately 120 kHz. At this clock frequency, the execution time of about 2.5 ms (cf.

Key byte	1	2	3	4	5	6	7	8
# Required traces	700	4,400	3,300	200	4,100	4,200	4,300	2,200
Key byte	9	10	11	12	13	14	15	16
# Required traces	2,800	2,100	2,300	4,500	1,400	1,100	3,100	500

Table 9.1: Approximate number of required power traces to recover respective bytes of the AES key. Metric: Ratio between correlation for correct key candidate and second highest correlation greater than 1.1

Fig. 9.9) would imply that the AES is performed in only 300 clock cycles. Considering that even highly optimized AES implementations require about 3,000 cycles on similar (and probably more powerful) 8-bit µCs [BOS09], it appears that the leakage is distributed over several clock cycles, presumably due to the lowpass characteristic mentioned in Sect. 9.4. Hence, we continued our analysis using the EM traces that give a higher temporal resolution in this regard.

9.5.2 Key Recovery using EM Traces

We performed the identical attack as in Sect. 9.5.1 on the (digitally demodulated) EM traces. The resulting correlations after 800 traces for all candidates for the first, second, eighth, and ninth key byte are shown in Fig. 9.14. In contrast to the power traces, the correlation for the correct key candidate clearly exceeds the one for the wrong candidate after already less than 1,000 traces. Besides, the correlation peak is limited to a short instant of approximately 160 ns, which corresponds to a clock frequency of about 6.25 MHz. Thus, it is likely that this correlation is for one or a few instructions of the µC only.

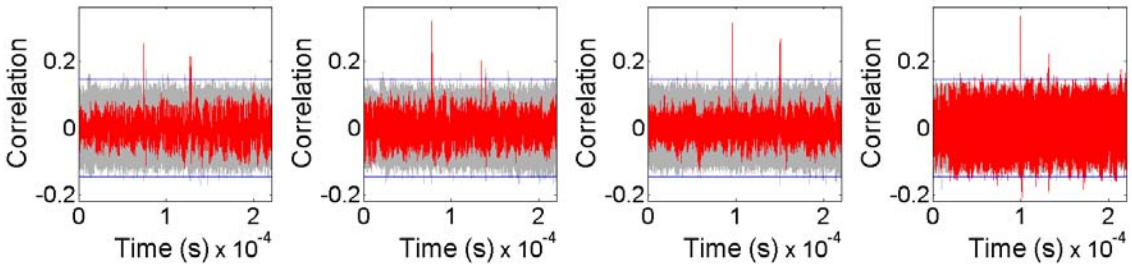


Figure 9.14: Correlation coefficient for all candidates for the key bytes 1, 2, 8, and 9 (left to right) after 800 EM traces. Red: correct key candidate, gray: wrong key candidates

Again, we estimated the number of required traces to recover respective key bytes in analogy to Sect. 9.5.1. The results are given in Tab. 9.2. Figure 9.15 shows the evolution of the maximum correlation and was used to derive the numbers given in Tab. 9.2.

As evident in Tab. 9.2, a maximum number of 500 traces is sufficient to fully recover the 128-bit AES key. Due to approximately 25 % of the EM traces being unusable, this translates to an overall number of 666 traces. Thus, only 1 h of access to the Yubikey 2 is sufficient to recover the key with EM measurements, compared to 10.5 h that would be required when using the power traces.

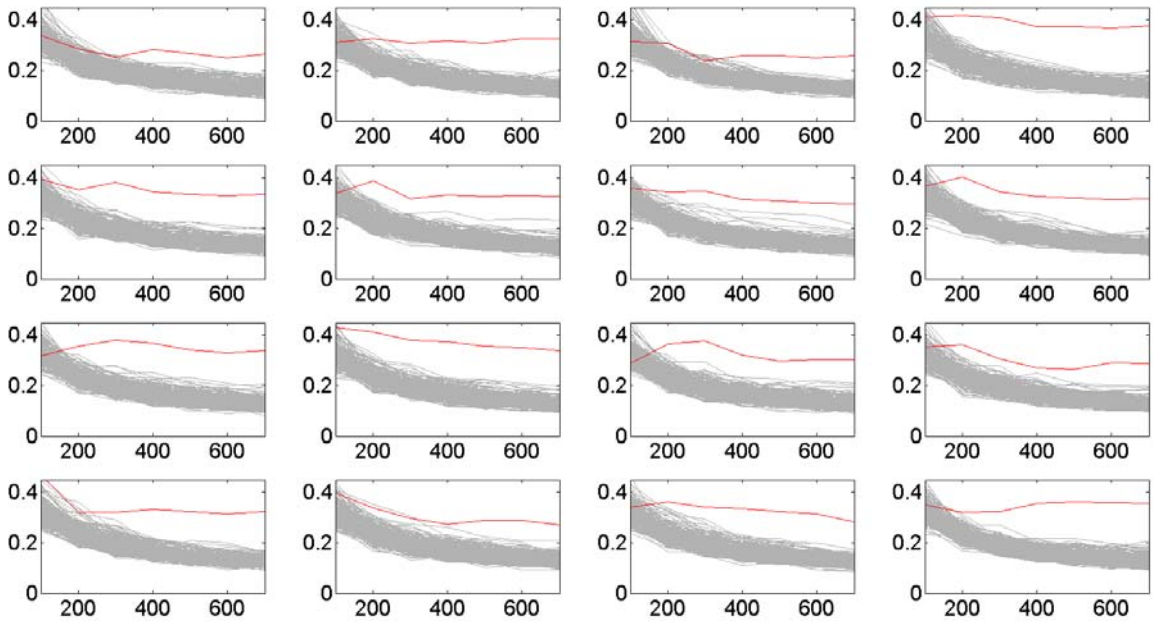


Figure 9.15: Evolution of the maximum correlation (vertical axis) for EM measurements over the number of used traces (horizontal axis) for all key bytes (left to right, top to bottom). Red: Correct key candidate

Besides, a tradeoff between computation time and the number of traces could be applied. An adversary may, for instance, decide to only record 300 traces, so three key bytes (1, 3, and 14) would not be (fully) recoverable. However, these remaining three bytes, i.e., 24 bit, could be easily determined using an exhaustive search on a standard PC within minutes. In this case, the measurement time is reduced to 36 min for effectively 400 traces in total.

9.6 Conclusion

Using a non-invasive side-channel attack, we are able to extract the full 128-bit AES key stored on a Yubikey 2 with approximately 500 EM traces. With the AES key, an adversary is able to generate an arbitrary number of valid OTPs and thus to impersonate the legitimate owner given

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

Table 9.2: Approximate number of required traces to recover respective bytes of the AES key using EM traces. Metric: Ratio between correlation for correct key candidate and second highest correlation greater than 1.1

that the traditional credentials have been obtained, e.g., by means of eavesdropping, phishing, or malware. To acquire the required number of traces, an adversary needs less than one hour of physical access to the Yubikey. Thus, the attack could for instance be carried out during the lunch break.

The attack leaves no physical traces on the DUT. The only means by which the attack could be detected is a (relatively high) increase of the usage counters, cf. Sect. 9.2.2. Due to the fact that the volatile session counter has to reach 256 first before the non-volatile usage counter is incremented, the EM-based attack only increases the usage counter by two when recording 500 traces. Thus, the presented attack does not lead to a "suspicious" change of this counter and is very unlikely to be detected in this way.

From a general point of view, the results of this chapter demonstrate that a different measurement method can drastically improve an SCA attack. Without having discovered the EM leakage, the DUT would have been considered to be significantly more secure. The use of power traces would have limited the practical impact of the attack, as the relatively long time of 10.5 h for the data acquisition increases the probability for the adversary being detected. For a system designer, this outcome implies that an implementation must be examined with respect to several attack vectors—solely relying on the analysis of one or a few possible attacks can lead to a false sense of security.

To mitigate the consequences of the attack described in this chapter, countermeasures both on the hardware level and for the (organization of the) backend should be implemented. In this regard, as part of the process of responsible disclosure, we discussed feasible approaches with the vendor Yubico. In general, the Yubikey should of course always be treated as a second factor and never be used as the sole means of authentication. Secondly, it should be ensured that no two Yubikeys have the same AES key. Otherwise, obtaining the AES key from one device would render all other devices with the same key insecure as well. Using only the 6-byte private ID mentioned in Sect. 9.2.2 to distinguish Yubikeys is hence not advisable in our opinion. Besides, especially for sensitive applications, users should be trained to keep their Yubikey with them at all times and report lost or stolen devices instantly so that they can be blocked and replaced.

On the level of the hardware and embedded software of the Yubikey 2, specific countermeasures against SCA can be realized to considerably increase the number of required traces, cf. Sect. 12.5. This in turn would reduce the threat posed by the attack: the longer the device has to be in the hands of the adversary, the more likely it is that the attack is noticed by the legitimate user. However, due to the limitations of the 8-bit μC used on the Yubikey 2, it is unclear whether SCA countermeasures such as masking that involve a higher space and time overhead can be implemented.

One interesting alternative—especially for high-security applications—is the Yubikey Neo also produced by Yubico [Yub13d]. Instead of a standard μC, the Yubikey Neo employs a CC certified smartcard controller that was specifically designed to withstand implementation attacks and thoroughly tested in this regard. In our opinion, to protect sensitive services and data, the price doubling of USD 50 compared to USD 25 for the Yubikey 2 may be a reasonable investment.

9.7 Reaction of the Vendor

Having discovered the security problem, before publication, we contacted the vendor Yubico as mentioned before. Yubico acknowledged our results and has taken measures to mitigate the security issues. We examined an updated firmware (version 2.4) and found that our attacks do not apply to this improved version. Several attempts to circumvent the new mechanisms implemented by the vendor were unsuccessful. Thus, the resistance of the DUT against SCA seems to have increased significantly. This likely rules out low-complexity attacks (in terms of the equipment and the required time for the measurements) as presented in this chapter. The following statement summarizes the reaction of the vendor Yubico:

> "Yubico takes security seriously and we welcome analysis of our products, and are happy to engage on a technical basis for the benefit of our customers. While the YubiKey Standard was not intended to resist physical attacks, we aspire to exceed expectations. After being informed about preliminary results, we worked with the research team to implement mitigations. We have incorporated this in our currently manufactured product. We wish to stress that the YubiKey NEO and the YubiKey Standard used in OATH or challenge response mode is not affected. We look forward to continue work with researchers and improve our products."

Chapter 10

Maxim SHA-1 Product Authentication ICs

The DS2432 and DS28E01 are wide-spread examples of a SHA-1 authentication IC product line manufactured by Maxim Integrated. Using a 64-bit secret, the ICs employ the SHA-1 in an HMAC-like construction to protect against product counterfeit: the chip is placed on a device (e.g., an ink cartridge or a mobile phone battery) and proves the authenticity to a host. However, presumably due to the low-cost nature of these ICs, countermeasures against implementation attacks were not incorporated. We show that a non-invasive side-channel attack using the power consumption of the devices can be used to extract the full 64-bit secret with approximately 2,000 traces for the DS2432 and approximately 2,500 traces for the DS28E01, requiring 40 min and 50 min of measurement, respectively.

Contents of this Chapter

10.1 Introduction

Counterfeit (electronic) products have become an immense problem for manufacturers and consumers. According to a report of the United Nations Office on Drugs and Crime [Uni13], the market for counterfeit goods had a value of USD 250 billion in 2012. Approximately 8% of all counterfeit products are electrical or computer equipment (based on the number of counterfeit seizures made at the European borders in 2008). Hence, protecting products against being "cloned" seems to be a necessity for a manufacturer today.

Devices that consist of several components of different complexities appear to be a profitable target for counterfeit in particular: For example, while fake printers are relatively rare, there is a huge variety of compatible ink cartridges for all brands, presumably because cartridges are easy to produce on the one hand and in constant demand on the other. The same holds for similar

low-cost items like accessories for mobile phones (e.g., chargers, batteries, etc.) and also for more expensive equipment like medical sensors or extension modules for network infrastructure.

To ensure that such (extension) components are genuine, today, several commercial solutions based on cryptographic authentication are available. Usually, an additional IC is placed on the device to be authenticated. The "host" (e.g., a printer or a mobile phone) then executes an authentication protocol with the IC to verify that the component (e.g., an inkjet cartridge or a battery) is genuine.

In this chapter, we focus on the SHA-1 EEPROM product line of Maxim Integrated, analyzing two specific ICs, the DS2432 and the DS28E01-100 [Max13]. These devices enable the (unilateral) authentication of a component to the host using a shared 64-bit secret in a challenge-response protocol based on SHA-1 as the central cryptographic primitive.

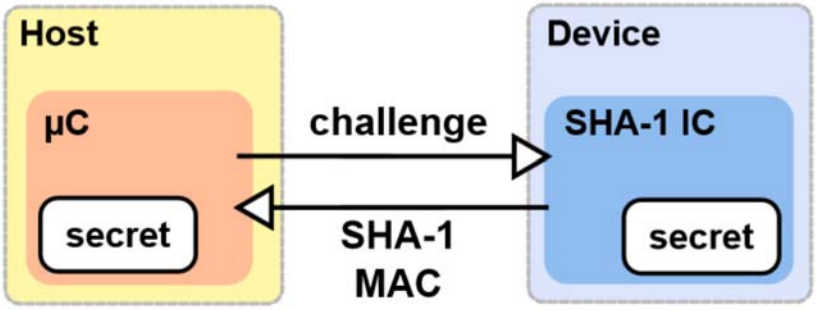


Figure 10.1: System for authenticating a device to a host using a SHA-1 IC by Maxim Integrated

The typical structure of the authentication system is depicted in Fig. 10.1: A µC on the host communicates with the SHA-1 IC on the device over a serial link and executes the challenge-response protocol. On a successful authentication (i.e., if the device can prove the knowledge of the shared secret), the device is regarded as genuine, otherwise, it is classified as counterfeit and further use is denied. For example, a printer may reject counterfeit ink cartridges, a smartphone does not boot when powered from a fake battery, and so on.

To assess how well the DS2432 and the DS28E01 are protected against physical attacks, we performed both non-invasive SCA (targeting the device's power consumption) and FI via manipulation of the supply voltage. We reported the results presented in this chapter to the vendor Maxim Integrated and are currently discussing possibilities to improve the protection offered by the devices against SCA.

10.1.1 Related Work

Implementation attacks on SHA-1 implementations have to our knowledge so far mostly been proposed theoretically: In [LLG09], a fault attack on the SHA-1-based cipher SHACAL-1 is proposed, which is extended to also apply for a standard SHA-1 Hash-based Message Authentication Code (HMAC) in [HH11]. With respect to SCA, McEvoy et al. described a CPA on their own implementation of a SHA-1 HMAC on an FPGA [MTMM07], also covering suitable countermeasures against this type of attack.

In a presentation at the 27th Chaos Communication Congress [Bra10], a sophisticated FI-based attack on an older SHA-1 device, the Dallas iButton, is described. The author also

discloses information on the used authentication protocol, which is similar to that of our DUTs. The attacks of [Bra10] are based on only partially overwriting the secret (which is achieved using FI) and may also apply to the DS2432 or the DS28E01 analyzed in this section, however, could be rather easily prevented by setting the corresponding write-protect flag for the memory storing the secret.

10.1.2 Contribution

In this chapter, as the main contribution we demonstrate that non-invasive SCA can recover the full 64-bit secret of both the DS2432 and the DS28E01 with approximately 2,000 and 2,500 traces, respectively, requiring 40 min and 50 min of measurement. To this end, we first describe the authentication protocol as used by Maxim Integrated in Sect. 10.2. We then outline a side-channel attack to fully recover the partially unknown input to the SHA-1 core function in Sect. 10.3. In order to practically apply this attack to the DUTs, we first profile the leakage characteristics in Sect. 10.4. We then show in Sect. 10.5 that the full 64-bit secret can be extracted from the DUTs. Besides, in Sect. 10.6, we give initial results for FI performed by manipulating the supply voltage. We conclude in Sect. 10.7, discussing implications and possible countermeasures.

10.2 Authentication Protocol

In order to perform an SCA, the initial step was to understand and implement the protocol used for authenticating the DUT to the host. We found the full datasheet for the older DS1961S iButton (which implements a similar SHA-1-based protocol) and source code for the communication with the DS2432 on the Internet. In this section, we describe the parts of the protocol relevant for an SCA.

On the electrical level, both the DS2432 and the DS28E01 employ a single line that is at the same time used for supplying the operating voltage and communicating with the host. Thus, the devices require only two connecting wires to the host (one for ground, one for data/supply voltage), which makes the ICs cost effective for mass products. The communication protocol supports bit rates of 15.3 $\mathrm{kBit/s}$ (“regular speed”) and 125 $\mathrm{kBit/s}$ (“overdrive speed”), whereas each bit is sent in a separate time slot. The duration for which the host pulls the data line low determines whether a one or zero is sent to the DUT. Similarly, for reading data from the DUT, the host begins a “read” slot and then disconnects its driver to bring the data line into a high-impedance state. To send a zero, the DUT then pulls the data line low for a defined period of time, otherwise, for a one, the data line is left at a high level.

To establish the authenticity of a Maxim SHA-1 IC, the host writes a 5-byte (for the DS28E01) or 3-byte (for the DS2432) challenge to the “scratchpad” memory of the device and issues a `ReadAuthPage` command. The DUT then computes the one-block SHA-1 hash over the data stored in the respective memory page, the UID of the DUT, the challenge, some constant values, and a 64-bit secret k^{dut}.

More precisely, let k_i^{dut} $(0 \leq i \leq 7)$ denote the bytes of the secret, P_i, $(0 \leq i \leq 31)$ the bytes of the read page of the DUT’s memory, M a value depending on the page address, ID_i, $(0 \leq i \leq 6)$ the bytes of the UID, c_i, $(0 \leq i \leq 4)$ the bytes of the challenges, and x_i, $(0 \leq i \leq 10)$ some (fixed constants). Then, the 512-bit (sixteen four-byte words) input to the SHA-1 is constructed

as shown in Tab. 10.1 for the DS28E01. For the DS2432, the input is almost identical, only the first two challenge bytes c_0 and c_1 are set to `0xFF` (because the challenge has a length of three byte for this device).

Word	Byte 3	Byte 2	Byte 1	Byte 0
0	k_0^{dut}	k_1^{dut}	k_2^{dut}	k_3^{dut}
1	P_0	P_1	P_2	P_3
2	P_4	P_5	P_6	P_7
⋮		...		
8	P_{28}	P_{29}	P_{30}	P_{31}
9	c_0	c_1	x_0	x_1
10	M	ID_0	ID_1	ID_2
11	ID_3	ID_4	ID_5	ID_6
12	k_4^{dut}	k_5^{dut}	k_6^{dut}	k_7^{dut}
13	c_2	c_3	c_4	x_2
14	x_3	x_4	x_5	x_6
15	x_7	x_8	x_9	x_{10}

Table 10.1: Input to the SHA-1 for the `ReadAuthPage` command of the DS28E01 (similar format for DS2432)

The SHA-1 is the computed according to the standard [NISa], except for the fact that only one block is hashed and the addition of the final $H_0^{(i-1)}$, ... is left out (specified in [NISa, p. 19, step 4]). Using the SHA-1 in this manner yields a simple HMAC, which may not be secure in "normal" applications (e.g., due to length-extension attacks [PO95]), but seems—at the first glance[1]—to be secure enough for the given application since only one block is hashed. Presumably, Maxim decided not to implement a standard HMAC [BCK96] to avoid invoking the SHA-1 two times.

Using the GIAnT (cf. Chap. 5), we implemented the complete protocol of the DS2432 and the DS28E01, also including the functions to set a specific secret or write to the memory of the DUT. We do not describe the details of these functions here, as they are not relevant for the actual SCA (but useful when profiling the device).

10.3 Theoretical Attack

In contrast to a typical block cipher, the SHA-1 round function depicted in Fig. 10.2 involves many linear operations, operates on 32-bit registers, and does not exhibit a clear division between varying input and constant (round) keys, so an "out-of-the-box" approach for SCA does not exist. Based on the SCA on a standard HMAC presented in [MTMM07], we thus developed an attack suitable for the way the SHA-1 is employed on the DUTs.

First, note that the output of the SHA-1 (after 80 rounds) is available to the adversary since it is the response of the DUT to the challenge of the host. Let this output be denoted as A_{80}, B_{80}, C_{80}, D_{80}, E_{80}, i.e., A_i is the value of register A after round i and so on. Then,

[1] We did not thoroughly analyze the mathematical security of the protocol

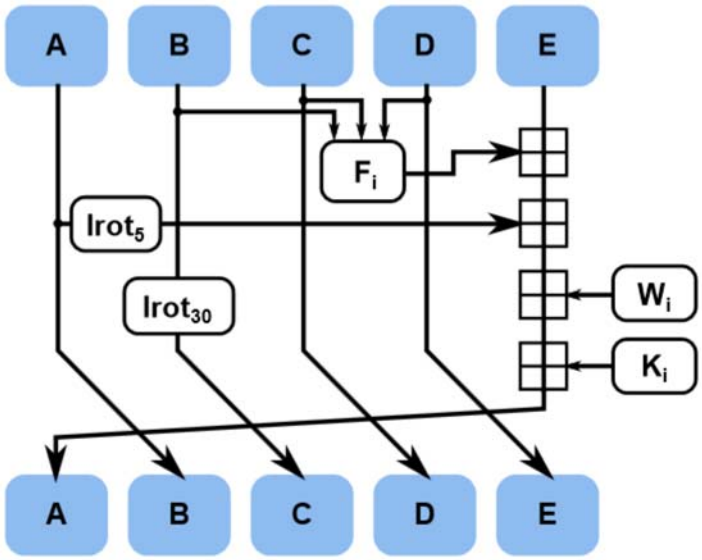


Figure 10.2: Round function of the SHA-1, updating the five 32-bit registers A–E. lrot_x denotes a left-rotate by x bit, $\boxplus$ addition modulo 2^{32}.

the four registers A–D after round 79 are given as $A_{79} = B_{80}$, $B_{79} = C_{80}$, $C_{79} = D_{80}$, and $D_{79} = E_{80}$. On the other hand, the remaining register E_{79} is computed as:

$$E_{79} = A_{80} - K_{80} - W_{80} - \text{lrot}_5\,(B_{80}) - F_{80}\,(C_{80},\, D_{80},\, E_{80})$$

The update involves the (unknown) value W_{80}, which depends both on the known parts of the input and the secret. However, note that the W schedule is linear (with respect to XOR):

$$W_i = \text{lrot}_1\,(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16})$$

Hence, W_{80} can be written as the XOR of a known (and eventually challenge-dependent) value W_{80}^{known} and an unknown value W_{80}^{secret} depending on k^{dut}. Thus, E_{79} can be determined by performing an SCA for all candidates for W_{80}^{secret}.

Basically, 2^{32} candidates would have to be considered, however, as proposed in [MTMM07], one can also consider partial correlations, i.e., first recover the lower eight bit of W_{80}^{secret}, then the lower 16, 24, and finally the full 32 bit of the secret value. After that, round 80 can be undone completely, and an identical attack is performed for round 79, 78, and so on. Having recovered the sixteen values $W_{80}, \ldots, W_{65}$, the W schedule can be completely inverted [HH11] and k^{dut} be determined.

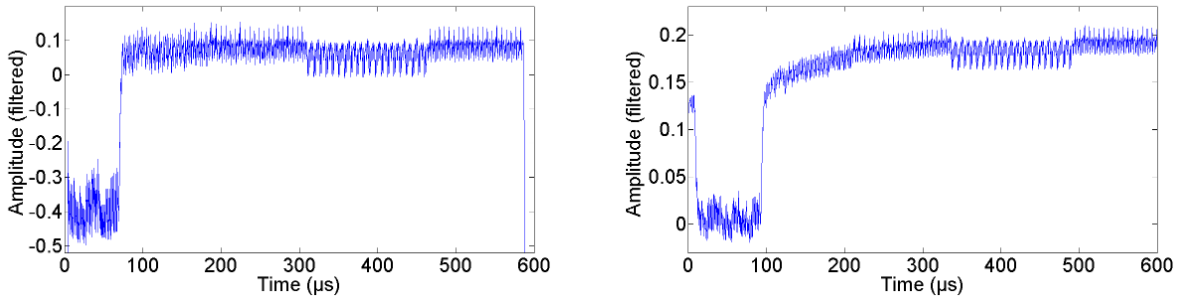
10.4 Side-Channel Profiling

In order to prepare practically performing the theoretical attack described in Sect. 10.3, we analyzed the leakage characteristics of the DUTs in a profiling phase. For the DS28E01, the secret was set to k^{dut} = `2d cf 46 29 04 b4 78 d8`, for the DS2432 to k^{dut} = `00 2d cf 46`

`29 04 b4 78`. With the known key, all intermediate values in the SHA-1 computation can be predicted during the profiling, e.g., to pinpoint the leakage of a specific round.

We used a standard SCA measurement setup, cf. Sect. 2.2.2, to record the power consumption of the DUTs. The ICs were soldered onto a prototyping PCB, with the two relevant pins (data and supply, ground) available on pin headers. The device was then controlled and powered using the GPIO pins of the GIAnT, cf. Sect. 5.2.2. A 50 Ω resistor was inserted into the ground path and the voltage drop over this resistor measured using the Picoscope 5204 at a sample rate of $f_s = 125$ MHz, cf. Sect. 3.2.1. We recorded 5,000 traces for each device, using uniformly distributed, random challenges. The traces were digitally lowpass-filtered with a cutoff frequency of 5 MHz, a value that we experimentally found to give the best results.

By visual inspection of several power traces, we found that the SHA-1 is executed after the host has read the 32-byte page data, one constant byte `ff`, and an (inverted) 2-byte CRC. After the SHA-1 has been computed, the host can read the generated HMAC as well. The SHA-1 computation appears in a trace as a region with increased overall power consumption, containing a sequence of 80 distinctive peaks that likely belong to the 80 rounds of the SHA-1 function. To illustrate this, Fig. 10.3 depicts average traces for the SHA-1 on the DS2432 and the DS28E01.



(a) DS2432

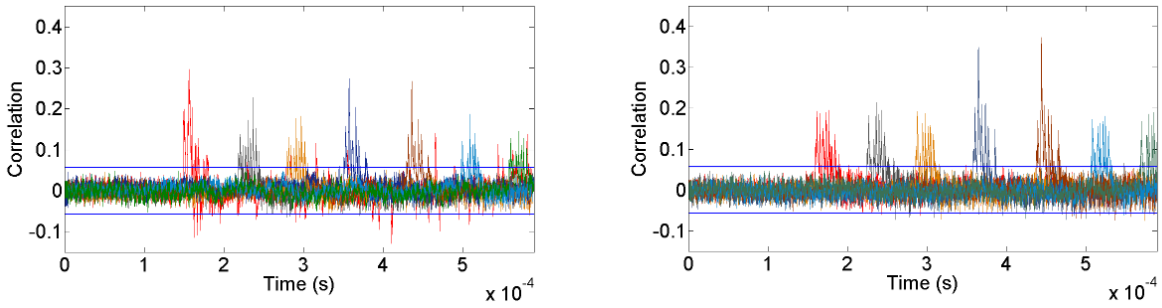
(b) DS28E01

Figure 10.3: Average (filtered) traces during the SHA-1 computation (computed using 5,000 traces)

For both DUTs, the trace begins with a drop of less than 100 µs. Our analysis indicates that in this region, presumably the SHA-1 engine is initialized and the input data is prepared. Then, the SHA-1 rounds are iteratively executed. Interestingly, the influence of the varying function F_i can be identified in the power trace: at around 300 µs (for the DS2432), the shape changes and the trace is slightly DC-shifted. This region corresponds to rounds 40 to 59, in which $F_i = (B \,\&\, C) \mid (B \,\&\, D) \mid (C \,\&\, D)$, with & the bitwise AND and | the bitwise OR. As the main difference between the two DUT, it can be seen that the DS2432 has a slightly higher power consumption and performs the SHA-1 computation approximately 10 µs faster than the DS28E01.

We the performed a CPA for all intermediate values of the SHA-1 registers A–E to find the time instants where the respective values are manipulated. It turned out that the HW is a

suitable power model for both DUTs[2]. Figure 10.4 depicts the correlation for the 32-bit HW of the register E after round 20, 30, 40, 50, 60, 70, and 80 after 5,000 traces.



(a) DS2432 (b) DS28E01

Figure 10.4: Correlation for the 32-bit HW of E after round 20, 30, 40, 50, 60, 70, 80 (peaks from left to right) after 5,000 traces

For both devices, significant correlation peaks can be observed with a maximum amplitude between approximately 0.2 and 0.3. These peaks occur for several clock cycles, which likely is caused by the fact that the value of E_i in round i equals D_{i-1} and thus occurs in 5 rounds as it is initially computed (as A) and then shifted from register to register.

10.5 Side-Channel Key Extraction

To extract the full 64-bit secret, we applied the theoretical attack described in Sect. 10.3, predicting bytes of E at the input of a round based to invert the round and obtain the value of W. We performed this attack for both the DS2432 and the DS28E01 for the last sixteen rounds of the SHA-1 and successfully extracted the secret from both DUTs.

To this end, Fig. 10.5 depicts the correlation coefficients for the lower 8, 16, 24, and the full 32 bit of E in round 80 for the DS28E01 after 5,000 traces. The red, dashed curve indicates the correlation for the correct candidate for the key-dependent part W_{80}^{secret}. Notice that the vertical scaling is different, because a higher correlation can be observed for predictions with more bits. Also note that since the targeted addition operation modulo 2^{32} is highly linear (with respect to XOR), many candidates exhibit a significant correlation, however, the correct candidate yields the maximum correlation and thus can be unambiguously identified. For the DS2432, a similar behavior can be observed.

To determine the minimum number of traces required to fully extract the secret, Fig. 10.6 depicts the evolution of the maximum correlation (for all candidates, correct candidate in red) over the number of used traces for the lower eight bit of E for both the DS2432 and the DS28E01. The predictions with more bytes (i.e., 16, 24, and 32) and the remaining rounds exhibit a similar or stronger leakage characteristic, so the number of required traces for round 80 approximately also holds for rounds 79–65.

[2] An unexpected discovery, because the DUTs contain a hardware engine for the SHA-1, a fact that usually implies a different model (e.g., HD)

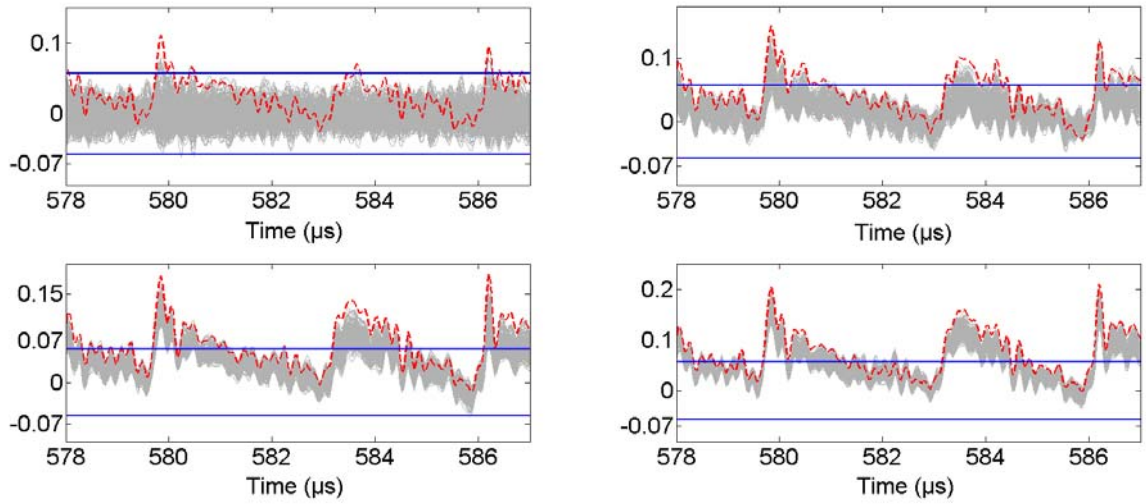
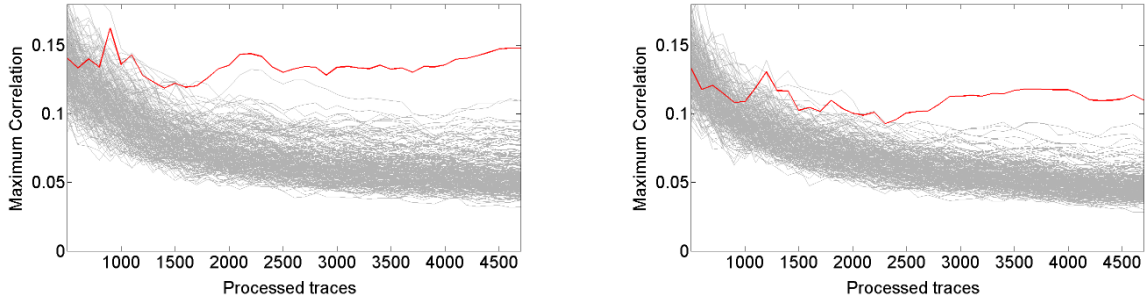


Figure 10.5: Correlation (vertical axis) for the lower 8, 16, 24, and the full 32 bit (left to right, top to bottom) of E in round 80 for the DS28E01 after 5,000 traces. Correct key candidate: red, dashed



(a) DS2432

(b) DS28E01

Figure 10.6: Evolution of the maximum correlation over the number of used traces for both DUTs. Red: Correct key candidate

As evident in Fig. 10.6, the correct candidate can be clearly distinguished from the wrong ones after approximately 2,000 traces for the DS2432 and approximately 2,500 traces for the DS28E01. As already mentioned in Sect. 10.4, it seems that the leakage of the DS2432 is slightly stronger: while for the DS28E01 the correct candidate reaches a stable correlation of approximately 0.11, for the DS2432 the maximum value is approximately 0.14. With our current setup, 1,000 traces can be acquired in approximately 20 min, translating to 40 min and 50 min to obtain the measurements required for extracting the full 64-bit secret of the DS2432 and the DS28E01, respectively.

10.6 Initial Results for FI

In addition to the SCA described in Sect. 10.5, we also analyzed the susceptibility of the DS2432 towards FI. Using the GIAnT (cf. Chap. 5), we generated a short pulse or glitch on the supply voltage during the computation of the SHA-1. We used a fixed key $k^{\text{dut}} =$ `00 2d cf 46 29 04 b4 78` and initially a constant challenge `ff ff ff` and could thus check whether a valid or a faulty HMAC was computed. The normal supply voltage of the DUT was set to 3.3 V.

We were able to disturb the circuitry of the DS2432 for a wide range of pulse widths t_{width}, offsets t_{offset}, and fault voltages V_{fault}. Injecting pulses with widths $t_{\text{width}} = 30 \ldots 50$ µs and fault voltages $V_{\text{fault}} = -7.625 \ldots 1.125$ V caused wrong results. An FI at the beginning or in the middle of the algorithm's execution yielded completely different HMACs for which it would be extremely difficult to identify the exact nature of the fault. Hence, we focused on faults at the end of the SHA-1. For example, for an offset (with respect to the transmission of the last bit before the SHA-1 starts) of $t_{\text{offset}} = 1.286$ ms, $t_{\text{width}} = 46$ µs, and $V_{\text{fault}} = 1.125$ V, the DUT returned the faulty HMAC (LSByte is sent first, i.e., the rightmost byte is the Most Significant Byte (MSByte) of A):

$$\overbrace{\texttt{f3 4c 28 0d}}^{\text{E}_{80}}\ \overbrace{\texttt{96 38 32 20}}^{\text{D}_{80}}\ \overbrace{\texttt{59 c6 88 a8}}^{\text{C}_{80}}\ \overbrace{\texttt{e9 b1 f4 fb}}^{\text{B}_{80}}\ \overbrace{\texttt{7a 86 4d 72}}^{\text{A}'_{80}}$$

whereas the correct, expected value is

$$\underbrace{\texttt{f3 4c 28 0d}}_{\text{E}_{80}}\ \underbrace{\texttt{96 38 32 20}}_{\text{D}_{80}}\ \underbrace{\texttt{59 c6 88 a8}}_{\text{C}_{80}}\ \underbrace{\texttt{e9 b1 f4 fb}}_{\text{B}_{80}}\ \underbrace{\texttt{7a 7e cd 6f}}_{\text{A}_{80}}$$

In this case, the fault only affected the update of the register A in round 80, changing `6fcd7e7a` (now denoted MSByte-first) into `724d867a`. While the XOR $A'_{80} \oplus A_{80} =$ `1D80F800` shows several affected bits, the difference $A'_{80} - A_{80} =$ `2800800` $\mod 2^{32}$ indicates that the fault likely led to three additional bits in one of the values added during the update of A being set.

To further investigate this issue, we fixed the parameters of the FI to the determined values and sent 5,000 uniformly chosen, random challenges. First, we noted that not in all cases a fault was successfully injected. Of 5,000 experiments, in 1,147 cases a faulty HMAC was returned, i.e., the success rate is approximately 23 %. For all faulty values, as expected, only the value of A was affected, however, not in a uniform manner: Table 10.2 gives ten challenges, the expected value for A_{80}, the obtained faulty A'_{80}, and the XOR and the difference of faulty and correct value.

As evident if Tab. 10.2, the observed faults likely affect a few bits during the update of A. However, the exact effect of a fault varied from experiment to experiment. We verified that even for constant input data and with the same pulse parameters, several different faulty values occurred: for example, for the challenge `ff ff ff`, we observed differences of `05000820`, `05080821`, `2800800`, and `00000800`.

Hence, without further information on the architecture of the IC, it seems difficult to directly exploit the FI for a working attack. Since we already demonstrated that a side-channel attack can be employed to extract the secret in 40 min for the DS2432, we did not further investigate the exact nature of the injected faults. However, the demonstrated susceptibility of the DUT suggests that a corresponding attack could be feasible and possibly further reduce the required time for extracting the secret.

Challenge	Correct A_{80}	Faulty A'_{80}	$A'_{80} \oplus A_{80}$	$A'_{80} - A_{80} \mod 2^{32}$
7e191b	0f7f3ce3	147f3ce3	1b000000	05000000
669594	a27bc9ea	63fc4a32	c18783d8	c1808048
5fbabf	e7ebf0dd	e7ebf13d	000001e0	00000060
09a370	06497667	084b8ecf	0e02f8a8	02021868
ae9a10	86e0d7ba	96e0d7ba	10000000	10000000
6c713e	d793c667	df95c667	08060000	08020000
d26989	b55e916a	b5deb172	00802018	00802008
9489fd	b1d303be	cdd303de	7c000060	1c000020
ae4c2a	13794836	12794836	01000000	ff000000
47b4f2	775887f3	77588ff3	00000800	00000800

Table 10.2: Correct and faulty SHA-1 outputs A_{80} for varying challenges

10.7 Conclusion

In this chapter, we demonstrated that the full 64-bit secret of the DS2432 and the DS28E01 SHA-1 authentication ICs can be extracted in 40 min and 50 min, respectively, using a non-invasive side-channel attack. To this end, we developed an analysis method suitable for the specific application of the SHA-1 in this scenario and successfully applied it to real measurements. This implies that an adversary could create an emulator behaving exactly like the original IC, a scenario especially conceivable in the context of product counterfeiting. Still, the adversary would have to build a custom emulating device to create an identical clone, because each genuine SHA-1 IC contains a factory-programmed UID that cannot be changed.

Thus, countermeasures should be taken: First, it should be made sure that every ICs has a unique secret, otherwise, an adversary having extracted the secret once may be able to duplicate all devices using this specific key. Thus, a suitable key derivation scheme involving the UID of the IC has to be implemented. One (simple) option would be to compute a standard HMAC of the UID, using a master key (only available on the host) as the key.

Besides, classical SCA and FI countermeasures should be employed, as discussed in Sect. 12.5 and [MTMM07]. For example, the timing of the algorithm could be randomized and masks be added to protect the intermediate values against SCA. Of course, this requires an entropy source of suitable quality and increases the complexity of the circuit. Thus, taking the low-cost application area of the SHA-1 ICs into account, a suitable trade-off between security and cost has to be found. We are in contact with the vendor Maxim Integrated and are discussing ways to prevent SCA-based attacks in future products.

Note that also the counterpart of the IC on the host should be protected: if the host uses a dedicated chip for verifying the HMAC, this IC has to protected in a similar manner—or even better, if, as suggested above, a master key for deriving device secrets is stored.

Chapter 11

Altera FPGA Bitstream Encryption

In order to protect FPGA designs against IP theft and related issues such as product cloning, all major FPGA manufacturers offer a mechanism to encrypt the bitstream used to configure the FPGA. From a mathematical point of view, the employed encryption algorithms, e.g., AES or 3DES, are highly secure. However, it has been shown that the bitstream encryption feature of several FPGA product lines is susceptible to side-channel attacks. In this chapter, we present the first successful attack on the bitstream encryption of the Altera Stratix II FPGA. To this end, we reverse-engineered the details of the proprietary and unpublished Stratix II bitstream encryption scheme from the Quartus II software. Using this knowledge, we demonstrate that the full 128-bit AES key of a Stratix II can be recovered by means of SCA with 30,000 measurements, which can be acquired in less than three hours. The complete bitstream of a Stratix II that is (seemingly) protected by the bitstream encryption feature can hence fall into the hands of a competitor or criminal—possibly implying system-wide damage if confidential information such as proprietary encryption schemes or secret keys programmed into the FPGA are extracted. In addition to lost IP, reprogramming the attacked FPGA with modified code, for instance, to secretly plant a hardware trojan, is a particularly dangerous scenario for many security-critical applications.

Contents of this Chapter

11.1 Introduction

In the field of digital design, FPGAs close the gap between powerful but inflexible ASICs and highly flexible but performance-limited µC solutions. FPGAs combine some advantages of software (fast development, low non-recurring engineering costs) with those of hardware (performance, relative power efficiency). These advantages have made FPGAs an important

component in embedded system design, especially for applications that require heavy processing, e.g., for routing, signal processing, or encryption.

Most of today's FPGAs are (re)configured with bitstreams, which is the equivalent of program code for FPGAs. The bitstream determines the complete functionality of the device. In most cases, FPGAs produced by the dominant vendors use volatile memory, e.g., SRAM, to store the bitstream. This implies that the FPGA must be reconfigured after each power-up. The bitstream is stored in an external Non-Volatile Memory (NVM), e.g., EEPROM or Flash, and is transferred to the FPGA on each power-up.

One of the disadvantages of FPGAs, especially with respect to custom hardware such as ASICs, is that an attacker who has access to the external NVM can easily read out the bitstream and clone the system or extract the IP of the design. The solution the industry provides to prevent this issue is a security feature called *bitstream encryption*. This scheme is based on symmetric cryptography in order to provide confidentiality of the bitstream data. After generating the bitstream, the designer encrypts it with a secure symmetric cipher such as the AES using a secret key k_{design}. The encrypted bitstream can now be stored in the external NVM. The FPGA possesses an internal decryption engine and uses the previously stored secret key k_{FPGA} to decrypt the bitstream before configuring the internal circuitry. The configuration is successful if the secret keys used for the encryption and decryption of the bitstream are identical, i.e., $k_{design} = k_{FPGA}$. Wire-tapping the data bus or dumping the content of the external NVM containing the encrypted bitstream does not yield useful information for cloning or reverse-engineering the device, given the adversary does not know the secret key.

11.1.1 Related Work

The cryptographic scheme used by Xilinx FPGAs starting from the old and discontinued Virtex II family to the recent 7 series is 3DES or AES in Cipher Block Chaining (CBC) mode [Kru04, Tse05]. Findings reported in [MBKP11] and [MKP12] show the vulnerability of these schemes to state-of-the-art SCA. It has been demonstrated that a side-channel adversary can recover the secret key stored on the target FPGA and use the key for decrypting the bitstream. Similar results have been reported for bitstream security feature of a family of Flash-based Actel FPGAs of Microsemi [SW12b].

11.1.2 Contribution

In this chapter, we analyze the bitstream protection mechanism called *design security* of Altera's Stratix II FPGA family. A detailed description of this real-world attack illustrating the steps required to perform a black-box analysis of a mostly undocumented target, i.e., the design security feature of the targeted FPGA family, is given. Similar to the attacks on the bitstream encryption of Xilinx and Actel FPGAs, our attack on the targeted Altera FPGA makes use of the physical leakage of the embedded decryption module. However, a detailed specification of the design security scheme is not publicly available. By reverse-engineering the Quartus II software application, we recovered all details and proprietary algorithms used for the design security scheme. Our results show the vulnerability of the bitstream encryption feature of Altera's Stratix II FPGAs to SCA, leading to a complete break of the security feature and the anti-counterfeiting mechanism.

This chapter's content is the result of joint work with Pawel Swierczynski and Amir Moradi. The attacks were presented at the FPGA 2013 conference and published in [MOPS13].

The remainder of this chapter is organized as follows. In Sect. 11.2, we describe the steps needed to reverse-engineer the Quartus II application in order to reveal the details of the design security scheme. Also, basic security problems of the according scheme are illustrated. The details of our side-channel attacks are presented in Sect. 11.3 and Sect. 11.4. Finally, in Sect. 11.5, we conclude, summing up our research results.

11.2 Reverse-Engineering the Design Security Scheme

For an SCA, all details of the bitstream encryption scheme are required. However, this information cannot be found in the public documents published by Altera. In this section, we thus illustrate the method we used to reveal the essential information, including the proprietary algorithms for the key derivation and the encryption scheme.

11.2.1 Preliminaries

The main design software for Altera FPGAs is called "Quartus II". To generate a bitstream for an FPGA, the Hardware Description Language (HDL) sources are first translated into a so called .SOF file. In turn, this file can then be converted into several file types that are used to actually configure the FPGA, cf. Tab. 11.1. For the purposes of reverse-engineering the bitstream format, we selected the .RBF type, i.e., a raw binary output file. This format has the advantage that it can be used with our custom programmer, cf. Sect. 11.3.1.

File extension	Type
.HexOut	Hexdecimal Output
.POF	Programmer Object File
.RBF	**Raw Binary File**
.TTF	Tabular Text File
.RPD	Raw Programming Data
.JIC	JTAG Indirect Configuration

Table 11.1: Bitstream file formats generated by Quartus II

For transferring the bitstream to the FPGA, Altera provides several different configuration schemes [Str07, p.131-132]. Table 11.2 gives an overview on the different available schemes. For our purposes, we used the Passive Serial (PS) configuration scheme, because it supports bitstream encryption and moreover, because the configuration clock signal is controlled by the configuration device.

Regarding the actual realization of the bitstream encryption, relatively little information is known. In the public documentation [AN309], it is stated that Stratix II uses the AES with a 128-bit key. Furthermore, a key derivation scheme is outlined that generates the actual encryption key given two user-supplied 128-bit keys. Apart from that, no information on the file format, mode of operation used for the encryption, etc. was initially available to us. Thus, in the following, we analyze the functional blocks of Quartus II and completely describe the mechanisms used for bitstream encryption on the Stratix II.

Mode	Bitstream Enc.
Fast Passive Parallel (FPP)	Yes
Active Serial (AS)	Yes
Passive Serial (PS)	**Yes**
Passive Parallel Asynch. (PPA)	No
JTAG	No

Table 11.2: Configuration modes for the Stratix II

11.2.2 RBF File Format

In order to understand the file structure of an .RBF file, we generated both the encrypted and the unencrypted .RBF files for an example design and compared the results. We found that the file can be divided into a header and a body section. Comparing the encrypted and the unencrypted .RBF files, we figured out that only a few bytes vary in the header. In contrast, the bodies containing the—possibly encrypted—actual bitstream are completely different. The unencrypted file's body contains mainly zeroes, while the encrypted file consists of seemingly random bytes.

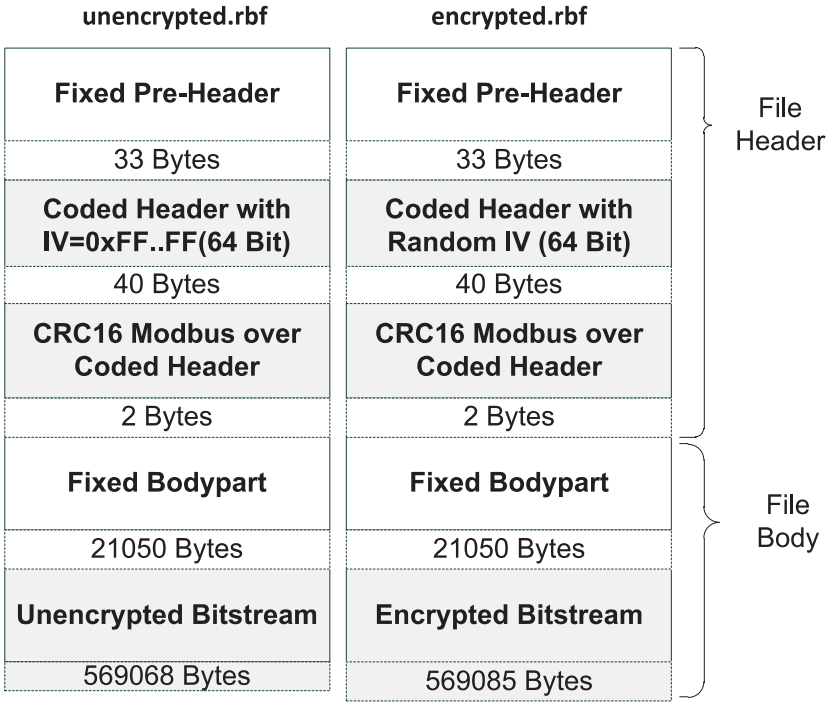


Figure 11.1: Structure of an unencrypted and an encrypted .RBF file

We encrypted the same input (.SOF file) twice, using the same key both times. It turned out that the resulting encrypted bitstreams are different, with differences in some header bytes and the complete body. Thus, the encryption process appears to be randomized in some way. Experimentally, we found that this randomization is solely based on the current state of the

system clock. Using a small batch script, we set the system clock to a fixed value and again generated two encrypted .RBF files. The resulting files were completely identical, confirming the conjecture that the PC clock is used as an Initialization Vector (IV) for the bitstream encryption.

To gain further insight into the internals of the file format, we used the reverse-engineering tool Hex-Rays IDA Pro [IDA]. This program allows analyzing the assembly code of an executable program (in our case the Quartus II bitstream tool) and run a debugger to display register values etc. while the target program is running. Using IDA Pro, we obtained the file structure depicted in Fig. 11.1 (for the specific FPGA fabric EP2S15F484C5N).

Both the unencrypted and encrypted .RBF files start with a fixed 33-byte "pre-header". The following 40 bytes include the IV used for the encryption. For the unencrypted file, the IV is always set to `ff...ff`, while for the encrypted file the first (left) 32-bit half is randomized (using the system clock). The right 32-bit half is set to a fixed value. However, the IV is not directly stored in plain; rather, the single bits of the IV are distributed over several bytes of the header. Using IDA Pro, we determined the byte (and bit) positions in the header at which a particular IV bit is stored.

Table 11.3 shows the resulting IV bit positions. The notation Y_{bitX} refers to bit X (big endian, $X \in [0, 7]$) of the byte at position Y in the .RBF file. Note that the byte positions are counted starting from the beginning of the .RBF file, i.e., including the fixed 33-byte pre-header.

IV bit	63	62	61	60	59	58	57	56
Position	49_{bit3}	48_{bit3}	47_{bit3}	46_{bit3}	45_{bit3}	44_{bit3}	43_{bit3}	42_{bit3}
IV bit	55	54	53	52	51	50	49	48
Position	57_{bit3}	56_{bit3}	55_{bit3}	54_{bit3}	53_{bit3}	52_{bit3}	51_{bit3}	50_{bit3}
IV bit	47	46	45	44	43	42	41	40
Position	65_{bit3}	64_{bit3}	63_{bit3}	62_{bit3}	61_{bit3}	60_{bit3}	59_{bit3}	58_{bit3}
IV bit	39	38	37	36	35	34	33	32
Position	33_{bit4}	72_{bit3}	71_{bit3}	70_{bit3}	69_{bit3}	68_{bit3}	67_{bit3}	66_{bit3}
IV bit	31	30	29	28	27	26	25	24
Position	41_{bit4}	40_{bit4}	39_{bit4}	38_{bit4}	37_{bit4}	36_{bit4}	35_{bit4}	34_{bit4}
IV bit	23	22	21	20	19	18	17	16
Position	49_{bit4}	48_{bit4}	47_{bit4}	46_{bit4}	45_{bit4}	44_{bit4}	43_{bit4}	42_{bit4}
IV bit	15	14	13	12	11	10	9	8
Position	57_{bit4}	56_{bit4}	55_{bit4}	54_{bit4}	53_{bit4}	52_{bit4}	51_{bit4}	50_{bit4}
IV bit	7	6	5	4	3	2	1	0
Position	65_{bit4}	64_{bit4}	63_{bit4}	62_{bit4}	61_{bit4}	60_{bit4}	59_{bit4}	58_{bit4}

Table 11.3: Mapping between the IV bits and the header bytes

Only the third and fourth bit of a byte are used to store the IV bits. The other bits of the header are constant and independent of the IV. We assume that these bits store configuration options, e.g., whether the bitstream is encrypted. The header is followed by a two-byte Modbus CRC-16 [CRC] computed over the preceding 40 header bytes for integrity check purposes.

The body starts with a 21050-byte block that is equal for both encrypted and unencrypted files. This block is followed by the actual bitstream (in encrypted or unencrypted form). The unencrypted bitstream has a length of 569068 byte. For the encrypted bitstream, 17 additional

bytes are added. This is due to the fact that for the encrypted format, several padding bytes are appended. For the purposes of our work, the details of this padding are irrelevant, as the additional block does not carry data belonging to the actual bitstream.

11.2.3 AES Key Derivation

In the publicly available documents, it is stated that the 128-bit AES key used for the bitstream encryption is not directly programmed into the Stratix II. Rather, two 128-bit keys denoted as KEY1 and KEY2 are sent to the FPGA during the key programming. These keys are then passed through a key derivation function that generates the actual "real key" used to decrypt the bitstream on the FPGA. The idea behind this approach is that if an adversary obtains the real key (e.g., by means of a side-channel attack), he should still be unable to use the same (encrypted) bitstream to program another Stratix II (e.g., to create a perfect clone of a product). Since the real key (of the second Stratix II) can only be set given KEY1 and KEY2, an adversary would have to invert the key derivation function, which is supposed to be hard. We further comment on the security of this approach in the case of the Stratix II in Sect. 11.2.3.

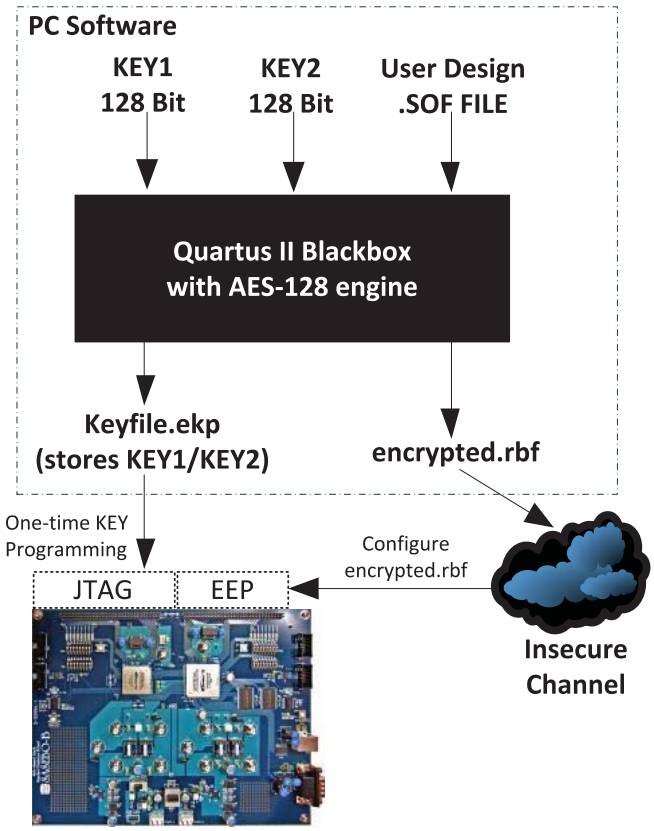


Figure 11.2: Quartus II black-box generating encrypted Stratix II bitstreams

Initially, the details of the key derivation were hidden in the Quartus II software, i.e., the software appeared as a complete black-box. As depicted in Fig. 11.2, Quartus II produces a key file (in our case Keyfile.ekp) that stores the specified KEY1 and KEY2. This key file is

later passed to the FPGA, e.g., via the Joint Test Action Group (JTAG) port using a suitable programmer.

However, the key derivation function is also implemented in Quartus II because the real key is needed to finally encrypt the bitstream. Hence, we again reverse-engineered the corresponding scheme from the executable program. Most of the cryptographic functions are implemented in the DLL file `pgm_pgmio_nv_aes.dll`. Apparently, the developers of Quartus II did not remove the debugging information from the binary executable; hence the original function names are still present in the DLL.

Figure 11.3 shows the corresponding function calls for the key derivation and the bitstream encryption. We focus on the key derivation, i.e., the upper part of Fig. 11.3. Note that due to the available debugging information, all function names are exactly those chosen by the Altera developers.

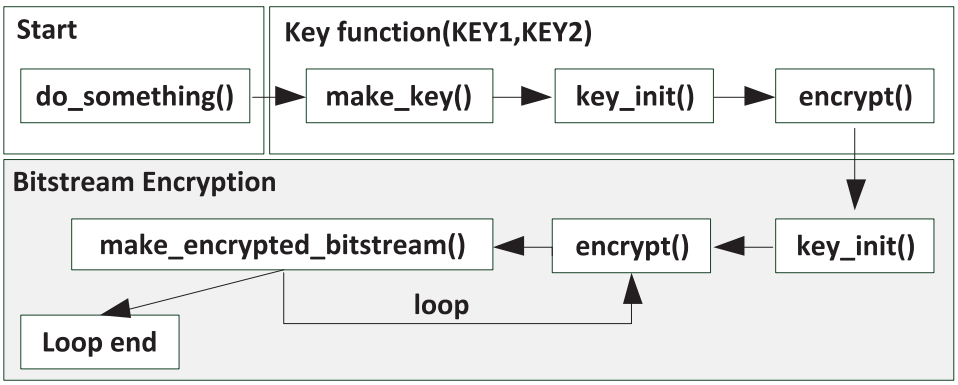


Figure 11.3: Quartus II call sequence during the bitstream encryption

First, the `do_something()` function checks the used key length. Then, the `make_key()` function copies the bytes of KEY1 to a particular memory location. The `key_init()` function then implements the key schedule algorithm of the AES, generating 160 byte of round keys in total. `encrypt()` then encrypts KEY2 with KEY1. Hence, the—previously unknown—key derivation function is given as

$$\textbf{Real Key} := \textbf{AES128}_{\textbf{KEY1}}(\textbf{KEY2}),$$

where KEY1 and KEY2 are those specified in the Quartus II application.

Worked Example

In order to further illustrate the details of the key derivation function, in the following we give the inputs and outputs for the chosen KEY1 and KEY2 we used for our analysis.

KEY1 (Quartus input, little endian)
`0x0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 01 00`

KEY2 (Quartus input, little endian)
`0x32 00 31 C9 FD 4F 69 8C 51 9D 68 C6 86 A2 43 7C`

Real Key = $\mathbf{AES128_{KEY1}(KEY2)}$ (big endian)
`0x2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C`

Security of the Key Derivation Function

At first glance, the approach of deriving the real key on the FPGA appears to be a reasonable countermeasure to prevent cloning of products even if the real key has been discovered. Yet, it should be taken into account that an adversary knowing the real key is still able to decrypt the bitstream and re-encrypt it with a different key for which he has chosen KEY1 and KEY2. Nevertheless, a product cloned in such a way could be still identified, because the re-encrypted bitstream will differ from the original one.

However, the way the AES is used for the key derivation in the case of the Stratix II does not add to the protection against product cloning in any way: a secure key derivation scheme requires the utilized function to be one-way, i.e., very hard to invert. For the Stratix II scheme, this is not the case. An adversary can pick *any* KEY1 and then decrypt the—previously recovered—real key using this KEY1. The resulting KEY2 together with KEY1 forms one of 2^{128} pairs that lead to the same (desired) real key when programmed into a blank Stratix II. The device will thus still accept the original (encrypted) bitstream, and the clone cannot be identified as such because KEY1 and KEY2 are never stored in the FPGA by design.

11.2.4 AES Encryption Mode

Having revealed the key derivation scheme, we focus on the details of the actual AES encryption, i.e., analyze the lower part of Fig. 11.3. First, the `key_init` function is executed in order to generate the round keys for the (previously derived) real key. Then, `encrypt()` is invoked repeatedly in a loop. Using the debugger functionality of IDA Pro, we exemplarily observed the following sequence of inputs to `encrypt()`:

```
0xB4 52 19 50 76 08 93 F1 B4 52 19 50 76 08 93 F1
0xB5 52 19 50 76 08 93 F1 B5 52 19 50 76 08 93 F1
0xB6 52 19 50 76 08 93 F1 B6 52 19 50 76 08 93 F1
...
```

Note that the first and the second eight byte of each AES input are equal. Moreover, this 64-bit value is incremented for each encryption, yielding (in this case) the sequence `B4`, `B5`, `B6` for the first byte. Apparently, the AES is not used to directly encrypt the bitstream. Rather, it seems that the so-called Counter (CTR) mode [NIS01] is applied. Figure 11.4 shows the corresponding block diagram.

In CTR mode, an IV is encrypted using the specified key. The output (i.e., ciphertext) of the AES is then XORed with the 16-byte data block to perform the encryption (of the bitstream blocks for the case of Stratix II). For each block, the IV is incremented to generate a new ciphertext to be XORed with the corresponding data block. The XOR operation is implemented in the function `make_encrypted_bitstream()`.

As mentioned in Sect. 11.2.2, the IV is generated based on the PC clock. Indeed, we found that the first four bytes of the IV correspond to the number of seconds elapsed since January 1,

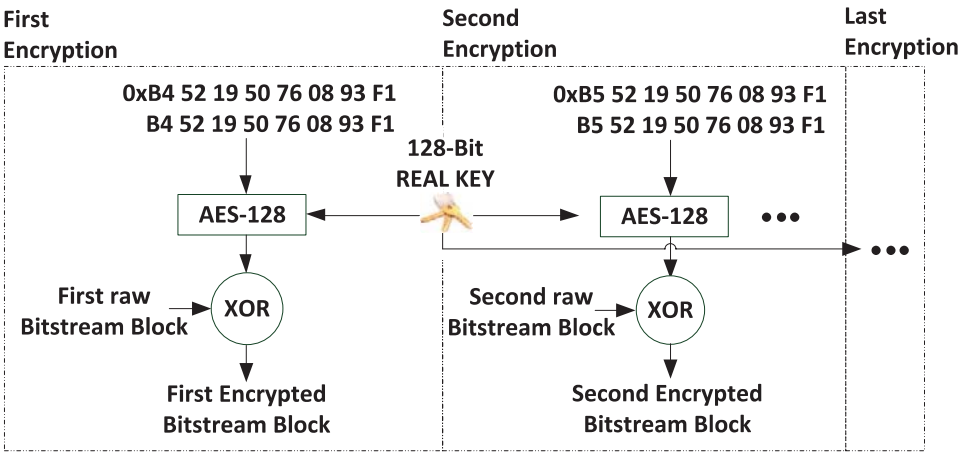


Figure 11.4: AES in CTR mode as used on Stratix II

1970. More concretely, the (little endian) value `0xB4 52 19 50` represents the date `2012.08.01 18:00:52`. The remaining four byte are constant. The overall structure of the IV is thus:

$$0x\underbrace{\mathbf{B4}\ 52\ 19\ 50}_{\text{Timestamp}}\underbrace{76\ 08\ 93\ F1}_{\text{Fixed bytes}}\underbrace{\mathbf{B4}\ 52\ 19\ 50}_{\text{Timestamp}}\underbrace{76\ 08\ 93\ F1}_{\text{Fixed bytes}}.$$

Having figured out the details of the AES key derivation and encryption, we implemented the aforementioned functions to decrypt a given encrypted bitstream. Given the correct real key and IV, we successfully decrypted the bitstream of an encrypted .RBF file. Figure 11.5 summarizes the details of the bitstream encryption process of Stratix II.

11.3 Side-Channel Profiling

With the knowledge of the bitstream encryption process presented in Sect. 11.2, we are able to analyze the Stratix II from a side-channel point of view. To this end, in this section, we first describe the measurement setup and scenario. Then, as a prerequisite to the according key extraction attack (Sect. 11.4), we apply SCA to find out the point in time at which the AES operations are executed.

11.3.1 Measurement Setup

Our DUT, a Stratix II FPGA (EP2S15F484C5N), is soldered onto a SASEBO-B board [AIS08] specifically designed for SCA purposes. The SASEBO-B board provides a JTAG port that allows one-time programming KEY1 and KEY2 into the DUT. For our experiments we set the real key to `2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C`, cf. Sect. 11.2.3.

We directly configured the DUT using the PS mode. For this purpose, we built an adapter that conforms to [Str07, p.599]. We developed a custom programmer based on an ATmega256 µC. Thus, we have precise control over the configuration process and are additionally able to set a trigger signal for starting the measurement process. This helped to record well-aligned

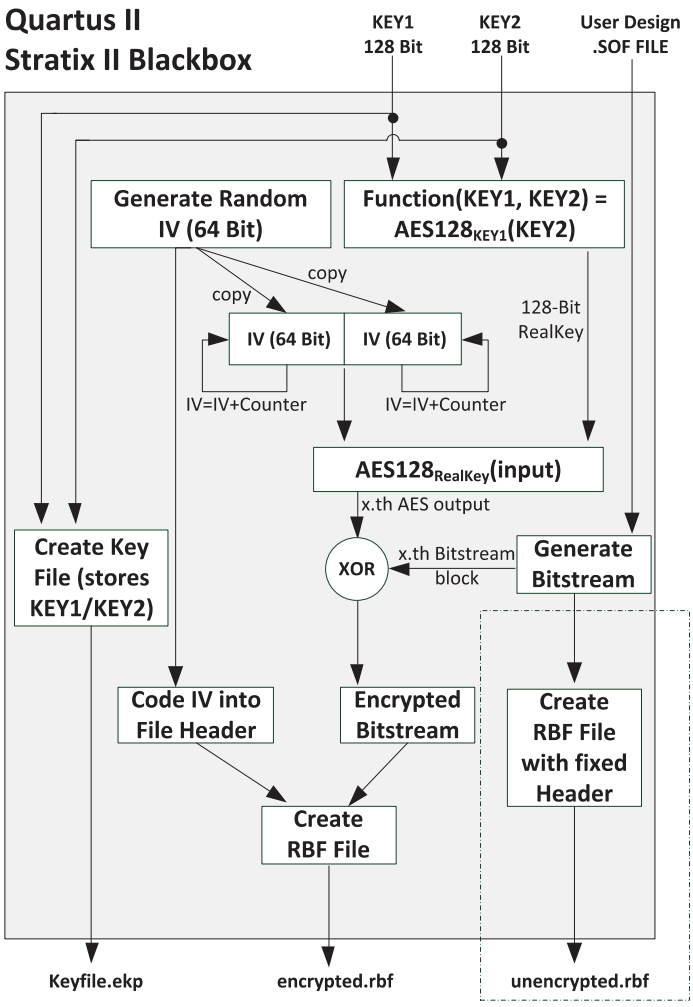


Figure 11.5: Overview of the bitstream encryption process for the Stratix II FPGA

power traces. Finally, our µC also provides the configuration clock signal to avoid (unwanted) internal clock effects that could, e.g., lead to clock jitter and therefore to misaligned traces.

According to [Str07, p.148], the DUT has three different supply voltage lines: V_{CCINT} (internal logic, 1.15 V–1.255 V), V_{CCIO} (input and output buffers, 3.00 V–3.60 V) and V_{CCPD} (pre-drivers, configuration, and JTAG buffers, 3.135 V–3.465 V).

For our analysis, we recorded the power consumption during the configuration of the DUT by inserting a small shunt resistor into the V_{CCINT} path and measuring the (amplified, AC-coupled) voltage drop using the LeCroy WavePro 715Zi DSO (cf. Sect. 3.2.1) as depicted in Fig. 11.6. We acquired traces with 225,000 data points each at a sample rate of 500 MHz. The respective (encrypted) bitstreams were generated on the PC built into the DSO and then sent to the DUT via the µC. The measurement process was triggered using a dedicated µC pin providing a rising edge shortly before the first bitstream block is sent.

During the decryption process of the encrypted bitstream, the AES is used in CTR mode. Hence, it might be possible that the DUT performs the first AES encryption when the header is

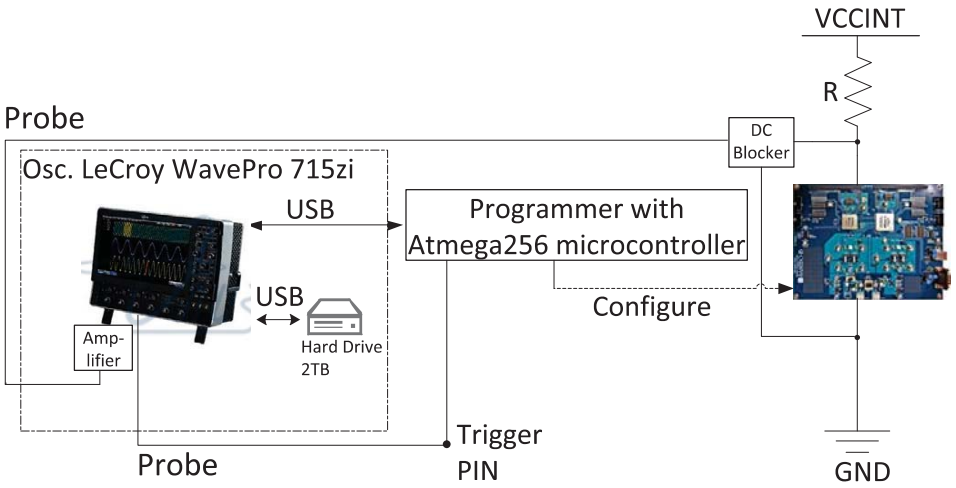


Figure 11.6: Measurement setup for the SCA of Stratix II

being sent because from that time onwards, the DUT knows the IV (first AES input). Therefore, we decided to perform a new power-up of the FPGA for each trace. The corresponding steps are described in more detail in Alg. 3.

Algorithm 3 Detailed steps for invoking the bitstream decryption on Stratix II and recording power traces

for i=1 to numberOfTraces **do**
  [µC] Perform DUT reset
  [µC] Transfer fixed 33-byte pre-header to DUT
  [PC] myIV[0..7] ← random value
  [PC] myHeader[] ← Get header from .RBF file
  [PC] Code myIV[] into myHeader[] (Table 11.3)
  [PC] Compute CRC-16 over coded header
  [PC] Send coded header with CRC-16 (42 byte) to µC
  [µC] Transfer coded header (42 byte) to DUT
  [µC] Transfer fixed body part (21,050 byte) to DUT
  [PC] Bitstream[0..47] ← random value
  [PC] Send Bitstream[] (48 byte) to µC
  [µC] Set trigger. Transfer bitstream (48 byte) to DUT
  [DSO] Record power trace
  [PC] Store trace i
  [PC] Store myIV[]
end for

11.3.2 Difference between Unencrypted and Encrypted Bitstream

Using our measurement script, we recorded 10,000 power traces for the time range that includes the transmission of 48 fixed, encrypted bitstream bytes. The FPGA decryption engine hence has the same input each time. In addition, we performed the same measurements while sending 48 byte of unencrypted bitstream. Finally, we computed the average power consumption over the set of our measured power traces, once for the unencrypted and once for the encrypted bitstream. Figure 11.7 illustrates the corresponding mean traces.

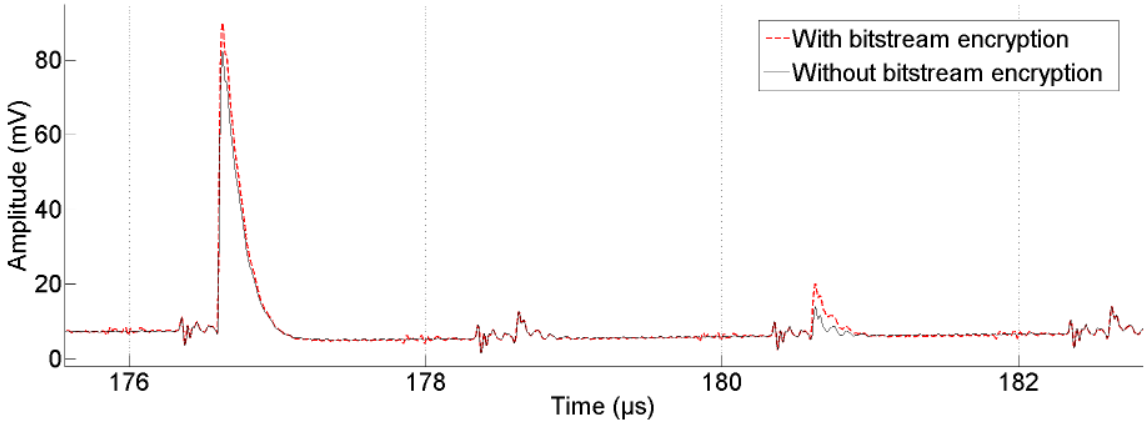


Figure 11.7: Average power consumption (10k traces) while sending an unencrypted (solid) and an encrypted (dashed) bitstream. Zoom on one byte.

As visible in Fig. 11.7, there is a significant difference in the average power consumption between the processing of the unencrypted bitstream and the encrypted bitstream. While the FPGA processes an encrypted bitstream, it consumes more energy compared to the processing of an unencrypted bitstream. A difference is already visible at the point where the first bitstream block is being transferred to the DUT. Thus, we assume that the AES encryption engine processes the first AES input (IV) while the programmer transfers the first encrypted bitstream block to the DUT. We further conjecture that while the programmer sends the second encrypted bitstream block, the DUT computes the XOR of the first AES output with the encrypted bitstream and configures the corresponding FPGA blocks.

11.3.3 Locating the AES Encryption

To verify our assumption on the correct time instance of the first AES encryption, we recorded another set of measurements and measured 840,000 power traces, this time exactly as described in Alg. 3. Then, for our profiling, we used the known key to compute all intermediate AES values for each IV challenge/trace.

For a CPA, we used this set of power traces to compute the correlation curves of about 220 different prediction models, e.g., each S-box bit of the first AES round, several HD models with different predicted register sizes, and several HW models for the intermediate AES states. As a result, the majority of our power models revealed a data dependency between the predicted power consumption and the measured traces. Hence, the FPGA evidently leaks sensitive information. Figure 11.8 shows the nine correlation curves for the states after each AES round.

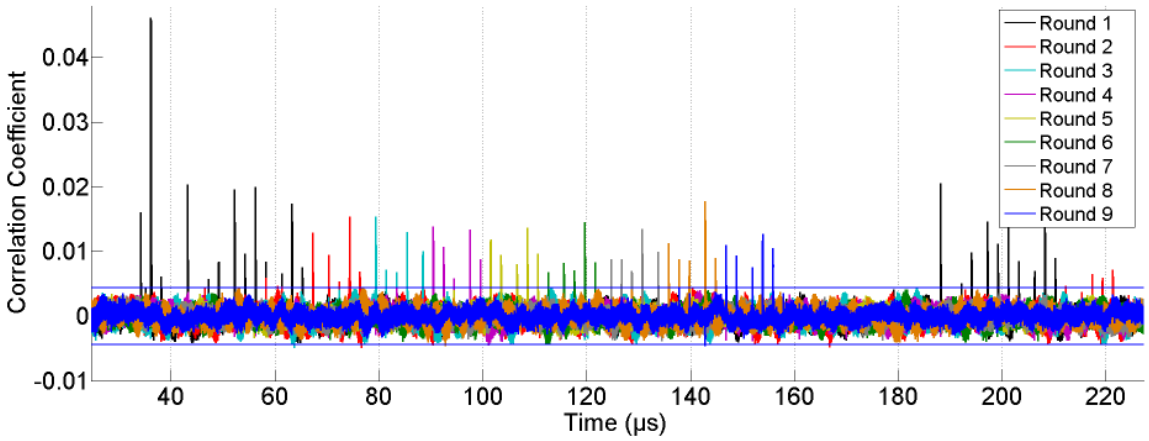


Figure 11.8: Correlation coefficient for one full `AddRoundKey` 128-bit state (one curve for each round). Utilized models: 1^{st} curve $\leftrightarrow$ HW of round 1, 2^{nd} curve $\leftrightarrow$ HW of round 2, etc.

The first correlation curve (black) that exhibits a peak up to an approximate value of 0.05 between 30 µs and 65 µs is for the HW model of the 128-bit state after the first round of the first AES encryption. The second correlation curve (red) is almost the same prediction model as before, but this time for the second round, and so on. Each round of the first AES encryption leaks and therefore, the correct time instance of the first AES encryption is located between 30 and 160 µs.

In Fig. 11.8, one can also spot the processing of the second AES encryption (starts at 180 µs). Due to the fact that only two byte of the IV are incremented each time, for the second AES encryption, the first output is similar to that of the first encryption. Therefore, the prediction of the first state of the first encryption automatically fits to the second as well. Thus, the same leakage (black curve in Fig. 11.8) appears for both the first and second AES encryption. Even the states after round 2 of both encryptions are slightly similar, and the leakage peak (red curve) appears for both encryption runs. Since the states (starting from round 3) are completely different for both encryptions, the predicted state of round ≥ 3 does not leak for the second encryption anymore.

11.4 Side-Channel Key Extraction

As shown in Sect. 11.3, the DUT exhibits a clear relationship between the power consumption and the internal states during the AES operation. In this section, we show how this side-channel leakage can be utilized to extract the full 128-bit AES key from a Stratix II with approximately three hours of measurements and a few hours of offline computation.

11.4.1 Digital Pre-Processing

As commonly encountered in SCA, the effect of the AES encryption on the overall power consumption is rather small (cf. Sect. 11.3). Hence, digital pre-processing of the traces to isolate the signal of interest (and thus reduce the SNR) can reduce the number of required

measurements, cf. Sect. 2.4. In the case of the Stratix II, we experimentally determined a set of pre-processing steps before performing the actual key extraction.

First, the trace is bandpass filtered with a passband from 500 kHz to 100 MHz. The, we applied DFA as described in Sect. 2.4.2. The signal was subdivided into windows of $w_{in} = 750$ sample points (i.e., 1.5 µs at the sampling rate of 500 MHz) with an overlap of 50% between adjacent windows. Each window was zero-padded to a length of 7,000 points. Then, the DFT of each window was computed and the absolute value of the resulting complex coefficients used as the input to the CPA. Note that we found the frequency with the maximum leakage to be around 2 MHz, hence, we left out all frequencies above 8 MHz to reduce the number of data points as well as the computational complexity of the CPA. Thus, each window (0...8 MHz) has a length of 112 points.

11.4.2 Hypothetical Architecture

For a side-channel attack to succeed, an adequate model for the dependency between the internal architecture and the measured power consumption is needed. In the case of the Stratix II, the internal realization of the AES was initially unknown. Hence, we experimentally tested many (common) different models as mentioned in Sect. 11.3.3. As a result, it turned out that the leakage present in the traces is best modeled by the HD *within* the AES state after the `ShiftRows` step [NISb]. More precisely, it appears that each column of the AES state is processed in one step, and that the result is shifted into a register, overwriting the previous column (that in turn is shifted one step to the right). The corresponding hypothetical architecture is depicted in Fig. 11.9.

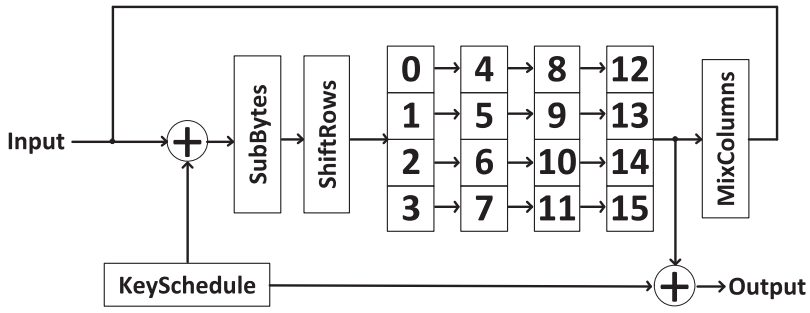


Figure 11.9: Hypothetical architecture of the AES implementation

For the key extraction in Sect. 11.4.3, we thus used the HD byte $0 \rightarrow 4$ (after `ShiftRows` in the first AES round) to recover the first key byte, byte $1 \rightarrow 5$ to recover the second key byte, and so on. As common in SCA, each key byte can be recovered separately from the remaining bytes, i.e., in principle 16×2^8 instead of 2^{128} key guesses for an exhaustive search have to be tested.

Note that, however, the initial state (i.e., the column overwritten with byte 0 ... 3) is unknown. Hence, we consider each row of the first two columns together and recover key byte 0 and 4, 1 and 5, 2 and 6, and 3 and 7 together, corresponding to 2^{16} key candidates each. After that, the remaining eight key bytes 8 ... 15 yield 8×2^8 candidates in total because the

previous (overwritten) column values are known. The total number of key candidates is thus $8 \times 2^8 + 4 \times 2^{16} = 264,192$ for which the CPA can be conducted within a few hours using standard hardware.

11.4.3 Results

Using the described power model, we computed the correlation coefficient for the respective (byte-wise) HD of the AES states. Figure 11.10 shows the result for the first S-box, i.e., the HD between byte 0 and 4. Evidently, the correct key candidate `0x2B` (solid, red curve) exhibits a maximum correlation of approximately 0.05 after 400,000 traces, clearly exceeding the noise level (horizontal blue lines) of $^4/\sqrt{\#\,\text{traces}} = 0.006$, cf. Sect. 2.3.2.

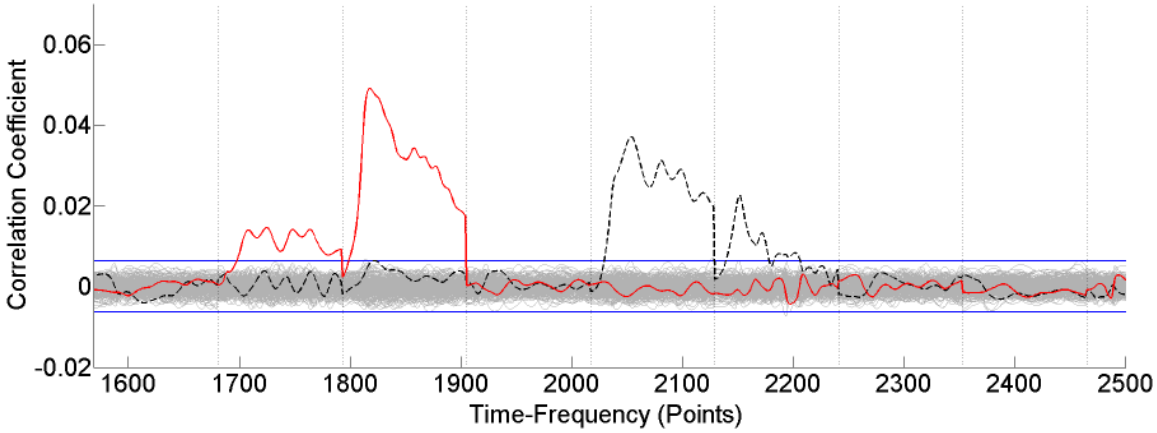


Figure 11.10: Correlation coefficient for the first S-box after 400k traces using DFT pre-processing. Correct key candidate `0x2B`: solid, red curve.

All other (but one) key candidates stay below the noise level. However, a second key candidate `0xAB` (dashed, black curve) also results in a significant peak at a different point in time. This is due to the fact that, as explained in Sect. 11.2, the first 64-bit half of the plaintext (i.e., the IV) equals the second half. Hence, a second key candidate (from the second 64-bit half) also exhibits a significant correlation. Indeed, the second peak (red) belongs to the correct key candidate `0xAB` for the corresponding key byte 8 in the second 64-bit half. As expected, due to the serial nature of the hypothetical architecture, the correlation occurs at a later point in time.

We conducted the CPA for all 16 AES S-boxes and obtained a minimal correlation coefficient (determining the required number of traces) of $\rho_{min} = 0.031$ for the fourth S-box. Hence, according to the estimation given in Sect. 2.3.2, the minimal number of traces to extract the full AES key is approximately $^{28}/\rho^2_{min} = 29,136$.

Figure 11.11 depicts the according correlation coefficient for the first S-box when leaving out the DFT pre-processing step. In general, the results are similar to those of Fig. 11.10, however, the observed correlation is halved compared to the CPA with the DFT pre-processing. Overall, we obtained a $\rho_{min} = 0.021$, i.e., 63,492 traces would be needed when leaving out the DFT pre-processing.

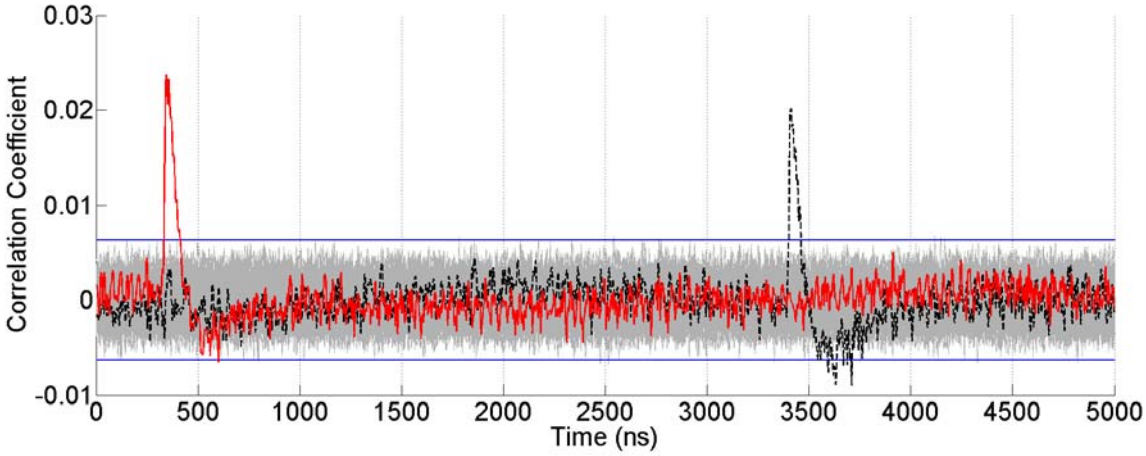


Figure 11.11: Correlation coefficient for the first S-box after 400k traces without DFT preprocessing. Correct key candidate `0x2B`: solid, red curve.

Using our current measurement setup, 10,000 traces can be recorded in approximately 55 min. Note that the speed of the data acquisition is currently limited by the µC; thus, the speed could be increased with further engineering efforts. Nevertheless, the amount of traces required to perform a full key-recovery can be collected in less than three hours.

11.5 Conclusion

After reverse-engineering the relevant functions of the Quartus II program, all details of the bitstream encryption, including the proprietary algorithms of the design security scheme have been revealed. Using this knowledge, a side-channel adversary can mount a successful key recovery attack on the dedicated decryption hardware. As a consequence of our attacks, cloning of products based on Altera Stratix II FPGAs protected with the bitstream encryption feature becomes possible. Moreover, an attacker can not only extract and reverse-engineer the bitstream, but might also modify it or create a completely new bitstream that would be accepted by the device. This threat is especially relevant in military applications, but could also have a major impact in other cases, e.g., for surveillance and trojan hardware scenarios. Furthermore, an unencrypted bitstream allows an adversary to read out secret keys from security modules or to recover proprietary cryptographic algorithms.

Part IV

Conclusion

Chapter 12

Attacks and Countermeasures

In this chapter, we summarize the steps typically required for performing an implementation attacks in practice. We then describe the central problems and flaws we came across in the course of our analyses. We compare the DUTs used in this thesis in terms of the number of required traces for a successful SCA, the respective time for acquiring the measurements, and the overall effort to conduct the attack. Based on this comparison and on a description of commonly made mistakes, we propose countermeasures on different levels to prevent according attacks or at least mitigate the consequences.

Contents of this Chapter

12.1 Steps of a Real-World Attack

In contrast to theoretical works on implementation attacks and SCA in particular, an attack on a real DUT involves additional steps that are usually not described in the literature. Hence, as an outcome of the case studies presented in Chap. 7–11, we describe the process of performing a real-world attack in this section.

12.1.1 Preparation

After a target device has been selected, the first step is to obtain as much information as possible on the DUT. Exact knowledge on the communication protocol and the cryptographic functionality is mandatory to successfully perform an implementation attack. In general, the more information is available, the less unknown factors remain that may cause an attack to fail otherwise. In order to not overestimate the security level of a DUT, it is essential to eliminate such factors and ensure that an analysis focuses on the actual protection against (implementation) attacks. For example, if a DUT uses an even slightly changed version of a standard cipher, an SCA will fail if the adversary is not aware of the modifications. However, the implementation on the DUT may still be susceptible to SCA once the details of the implementation are known.

Ideally, documentation, e.g., a datasheet, is available (as, e.g., for the Yubikey 2, cf. Chap. 9), however, in many practical cases such documents are kept confidential and not published. Sometimes, search engines can help to locate datasheets for the DUT or similar devices by the same vendor (e.g., for an older product generation) on the Internet. Besides, researchers may have already analyzed the DUT and for instance reverse-engineered the communication protocol. This was for example the case for the Maxim SHA-1 ICs (cf. Chap. 10) and the DESFire MF3ICD40 (cf. Chap. 7).

Otherwise, if no information can be obtained from public sources, the communication protocol and the (cryptographic) functionality has to be reverse-engineered. This step can be performed on (a combination of) different levels:

Protocols and Interfaces In the simplest case, observing the communication of the DUT, e.g., an RFID smartcard with a genuine reader, allows to retrieve the required details. In the past, this approach has been shown to be successful in certain cases, e.g., for Texas Instrument's DST [BGS+05]. However, for the case studies described in this thesis, this high-level approach turned out to be not suitable. For example, the frequent use of obscurity ruled out a protocol-only analysis of the SimonsVoss system described in Chap. 8. Even for Altera Stratix IIFPGAs—which use the AES in a relatively standard manner—reverse-engineering solely based on observing the inputs and outputs of the Quartus II software did not yield all details required for SCA, cf. Chap. 11.

PC Code In case PC software that implements the communication with a DUT is available, analyzing this program code using tool like IDA Pro [IDA] is usually the next step if protocol-only reverse-engineering fails. For the Altera Stratix II, this approach allowed to determine all details of the undisclosed cryptographic scheme and subsequently mount a side-channel attack. Likewise, we are aware of cryptographic RFIDs (not further described in this thesis) for which the proprietary communication protocols can be reverse-engineered from the PC software for a commercial reader.

Embedded Code When a DUT is based on a programmable device like a μC, (parts of) the embedded code can be examined using the same methods as for PC software. However, the embedded code is often protected (e.g., using a read-out protection) and hence not as readily available as PC code. Yet, the additional obstacle of obtaining the code can be overcome in many cases, e.g., by resetting the code protection fuse as described in Chap. 8. Besides, we also observed examples where programming interfaces like JTAG were not disabled and allowed direct access to the memory of the DUT.

Hardware In case of a hardware implementation, one may also attempt to infer the functionality using microscopic photographs of the silicon die. For the Mifare Classic RFID, circuit-level analysis led to the reverse-engineering of the proprietary Crypto1 cipher and enabled subsequent attacks [NESP08, GKGM+08, Cou09]. For the SimonsVoss locking system (Chap. 8), the schematic of the custom ASIC could be recovered in a similar manner. Although this did not directly result in a successful attack, it allowed to rule out the implementation of cryptographic functions on the ASIC.

Apart from having access to the information necessary to establish the communication, it is helpful to have several samples of the DUT available: Especially invasive attacks and FI may

lead to damaged or destroyed ICs. Besides, the general applicability of SCA can be shown by successfully extracting keys from several instances of the DUT. For all DUTs described in this thesis, we found sources to obtain the required samples: DESFire MF3ICD40 cards could be bought for approximately USD 3.30 on an Internet shop. The same holds for SimonsVoss components, where we also found several offers on online auctions websites. The Yubikey 2 can be easily ordered on the vendor's website. The Maxim SHA-1 ICs are available on IC marketplace websites, and for the analysis of Altera Stratix II we used a SASEBO-B board that was publicly sold.

12.1.2 Measurement

In order to prepare the lab setup for an implementation attack, the first step is to implement the communication with the DUT based on the previously obtained information as outlined in Sect. 12.1.1. For this purpose, an adversary may either use a standard off-the-shelf device, e.g., a commercial RFID reader or an FPGA programmer, or implement the communication protocol on his own. For the DESFire MF3ICD40, we for example utilized a custom RFID reader (cf. Sect. 3.1.2) to have full control over the authentication process. The GIAnT presented in Chap. 5 includes modules to flexibly implement (serial) communication interfaces and was, e.g., used for the analysis of the SimonsVoss lock (Chap. 8) and the Maxim SHA-1 ICs. For RF devices, an SDR like the USRP2 helps to reduce the time needed for establishing the communication compared to building application-specific RF circuits.

Being able to communicate with the DUT and thus to invoke the (cryptographic) operation to be analyzed, the actual physical side channel to be used can be selected. Usually, the two most common approaches—measuring the power consumption or the EM emanation (cf. Sect. 2.2)—are utilized. Depending on the complexity and the specific properties of the DUT, one method may be more suitable and thus be employed without testing the other.

For example, in the case of the DESFire MF3ICD40 (Chap. 7), opening the plastic package for direct access to the antenna would result in a permanent modification of the DUT and thus make the attack detectable afterwards. In contrast, for Altera Stratix II (Chap. 11) and the Maxim SHA-1 ICs, inserting a measurement resistor into the supply path was easily possible and hence used as the measurement method in these cases.

The example of the Yubikey 2 (Chap. 9) demonstrates that the utilized side channel may significantly affect the success rate of an SCA: while the current consumption traces are lowpass-filtered and cannot be precisely aligned, the EM traces allowed for precise alignment, thus reducing the number of required measurements from 7,000 to 666.

As shown by the example of RFID smartcards in Chap. 4, analog pre-processing of the side-channel signal (before it is digitized by the DSO) can help to isolate the signal of interest and increase the success rate of an attack. However, note that such filtering may also have the opposite effect, e.g., when filtering signal components containing the leakage.

In contrast to digital pre-processing, experimentally finding the "best" parameters for analog processing is more complicated since each parameters setting requires the acquisition of new traces and the physical modification of the filter. Hence, the use of analog pre-processing should be carefully prepared (e.g., by performing the processing digitally first) and compared to measurements without pre-processing to ensure that the signal quality is indeed improved.

12.1.3 Evaluation

For evaluating the acquired side-channel traces to extract cryptographic secrets, the first step is to profile the side-channel leakage of the DUT. This includes locating the points in time at which the targeted operation is executed, determining parameters like the clock frequency of the DUT, and the architecture of the implementation (synchronous or asynchronous circuit, µC or hardware implementation).

Based on the results of the profiling, further pre-processing in the digital domain may be applied, e.g., using a bandpass filter to isolate the frequency components around the DUT's clock frequency (as done for the SimonsVoss lock, cf. Sect. 8.4). As for analog pre-processing, the effects of this step should be compared to unprocessed measurements and the parameters adjusted accordingly. Optionally, one may also use a less heuristic approach and determine digital filter coefficients as proposed in Sect. 2.5.

For finally carrying out the actual key-extraction attack, several additional choices have to be made. First, the evaluation method has to be selected—in most cases, we used DPA or CPA in this thesis since these methods turned out to be generally applicable and do not require a separate profiling device as needed for TAs, cf. Chap. 2.

Subsequently, assumptions on the architecture of the implementation (to determine intermediate values to focus on) and the power model have to be made. In practice, exhaustively testing many possible architectures for a given algorithm combined with both HW and HD models is often the most direct way to find suitable choices for these parts of the attack. Whenever possible, these experiments should be conducted with a fixed, known key to be able to also predict values in later round of an algorithm and to reduce the computational complexity.

In addition, information obtained in the preparation step (Sect. 12.1.1) or during the profiling may help to reduce the number of possible implementations to consider. For example, if it is known that an algorithm runs on an 8-bit µC, assuming an byte-wise HW leakage is a reasonable starting point.

The ideal case is a fully known implementation, e.g., as for the SimonsVoss lock (Chap. 8) where the embedded code of the targeted cryptographic function was available. In this case, the appropriate intermediate values and the power model could be determined utilizing the outcomes of the preparatory reverse-engineering. Note that the observations made in the evaluation step may also require the repetition of the measurement step (Sect. 12.1.2) under modified conditions (e.g., changing the acquisition method, targeting a different part of a protocol, etc.).

Only when all mentioned prerequisites have been fulfilled, i.e., the evaluation method, the targeted intermediate value, and the power model have been selected, the actual key-extraction can be carried out. Again, depending on the results, repeating previous steps may be necessary or beneficial. For instance, in the case of the Yubikey 2, although we had a working attack in principle, going back to the measurement step and using the EM emanation instead of the power consumption led to improved results with a significantly lower number of required traces.

As the main result of this section, we emphasize that carrying out real-world side-channel attacks requires knowledge in different areas (e.g., electrical engineering, signal processing, and statistics) and involves several related steps. Solely focusing on the final key-extraction step and using SCA as tool for black-box analysis is unlikely to lead to a successful attack in practice. Instead, the numerous unknown factors that may cause an attack to fail should be determined in the described step-by-step manner.

12.2 Comparison of the Presented Attacks

In this thesis, several—in some aspects quite different—embedded devices were analyzed with respect to the susceptibility to SCA and related implementation attacks. Table 12.1 summarizes the key figures for the respective DUTs, including the targeted cryptographic algorithm, the utilized physical side-channel, the number of required traces, and the time consumed for the data acquisition.

DUT	Targeted algorithm	Measurement method	# required traces	Measurement time
DESFire MF3ICD40	3DES	EM	250,000	7 h
SimonsVoss Lock	Proprietary	EM	150	15 min
Yubikey 2	AES	EM	666	1 h
Maxim SHA-1 DS28E01	SHA-1	Power	2,500	50 min
Maxim SHA-1 DS2432	SHA-1	Power	2,000	40 min
Altera Stratix II FPGA	AES	Power	29,136	2.7 h

Table 12.1: Comparison of the side-channel attacks described in Chap. 7–11

Note that the acquisition times do not include the subsequent computations for further digital processing and the actual key extraction performed, e.g., using a CPA. However, in all cases, these *offline* computations could be carried out within a few hours on a standard PC. Moreover, using cloud-based services and parallelized implementations would further reduce the time needed for the evaluation phase. Thus, we consider the measurement time, i.e., the duration for which an adversary requires physical access to the DUT, as one of the most decisive factors for assessing the threat a specific SCA poses.

Due to the differences between the analyzed DUTs, the complexity and requirements of the respective attacks varied significantly. Overall, the Mifare DESFire MF3ICD40, originally sold as a high-security smartcard, required the highest amount of EM traces for a successful key extraction, cf. Chap. 7, and thus approximately seven hours for the data acquisition. Depending on the concrete application scenario of a DESFire, this may already rule out certain attacks. For instance, if the smartcard was used for access control and an adversary would (temporarily) steal a card to extract the keys and copy its access rights, there would be a high probability that the legitimate owner notices and reports the theft within seven hours. On the other hand, in a public transport scenario like that of the Opencard (Sect. 7.5), the card is permanently in the hands of a potential adversary. Thus, in this case, the time available for the extraction of secret keys and the duplication of cards is less limited.

For Altera Stratix II FPGAs, the DUT second-most difficult to attack, the key extraction (cf. Chap. 11) has more severe consequences. Recovering the secret key of *one* FPGA employed in a specific product (e.g., in a set-top box or a network router) allows an adversary to decrypt the complete bitstream defining the functionality of the device. This in turn enables other attack vectors, from the possibility to create an identical clone of a device to fully reverse-engineering a manufacturer's IP.

The DESFire MF3ICD40, the Altera Stratix II, and the Maxim SHA-1 ICs (Chap. 10) presumably utilize a dedicated part of their hardware to carry out the respective cryptographic operation (112-bit key 3DES, 128-bit key AES, 64-bit key SHA-1 HMAC). The remaining two DUTs, the Yubikey 2 (Chap. 9) and the SimonsVoss electronic lock (Chap. 8), employ 8-bit µCs to implement the cryptographic primitives, a 128-bit key AES and a combination of a proprietary function with the DES (with a key length of 128 bit in total), respectively. As expected, for the µC-based DUTs, a lower number of traces was sufficient to extract the secret key compared to hardware implementations.

As an interesting result, note that not only the "quality" of the cryptographic implementation determines the effort required to initially mount a side-channel attack. For several DUTs, a major problem was the lack of information on the employed protocol, the cryptographic primitives, and so on. The most significant example in this regard was clearly the SimonsVoss electronic lock described in Chap. 8. Since the system utilizes a proprietary and undisclosed obscurity function combined with a modified DES, an SCA was impossible until all details had been reverse-engineered. However, after this point, the strong leakage of the utilized PIC µC could be exploited to extract a system-wide key, leading to the attack with the lowest number of traces in this thesis.

A similar statement applies to Altera Stratix II FPGAs, where an undocumented key derivation based on the AES together with other obscurity measures required the reverse-engineering of the Quartus II design software. Again, with the access to all necessary information, carrying out a SCA attack was relatively straightforward. The same holds true—to a lower degree—for the Maxim SHA-1 ICs and the DESFire MF3ICD40. For both devices, the authentication protocol had been previously disclosed elsewhere. Without this information, the analysis would have required additional reverse-engineering. An example for the opposite situation is the Yubikey 2, for which the complete OTP computation scheme is publicly available. Thus, a significant amount of time could be saved during the analysis, as only a few further steps were needed for a full key-extraction.

In addition to SCA, we considered other implementation attacks on embedded devices. Using the GIAnT, we demonstrated examples for non-invasive FI by manipulating the supply voltage of a DUT. According attacks apply to most (unprotected) µCs, e.g., the Atmel ATXmega256 (cf. Sect. 5.3.1) and the Atmega163 (cf. Sect. 5.3.2) used in this thesis for verifying the functionality of the GIAnT, but also to PIC µCs [Osw09] and similar devices. For the hardware implementation of the SHA-1 ICs, FI techniques can be used to disturb the computations in the final round of the SHA-1.

Focusing on the physical layer of wireless devices in Chap. 6, we evaluated the active communication and passive eavesdropping ranges for 13.56 MHz HF and 433.92 MHz UHF RFID transponders in practice. In all cases, the achieved distances considerably exceed the specified values. Given that a system designer trusts in the specified ranges and plans an RFID installation accordingly, the increased ranges can lead to unexpected attack vectors that are completely independent of an eventually present cryptographic protection, e.g., relay attacks or the mere detection of the presence of a specific transponder.

All presented implementation attacks have in common that they can be carried out with low-cost equipment and without the need for a highly sophisticated lab setup. In almost all cases, a USD 3,000 Picoscope DSO was sufficient to acquire the side-channel traces. Only for the Altera Stratix II, we used a more expensive USD 25,000 LeCroy DSO, however, assume that the attack

could also be performed with a cheaper DSO. For controlling a DUTs and injecting faults on the supply voltage, we developed the open-source platform GIAnT presented in Chap. 5, which can be built for USD 300.

12.3 Reasons for Security Problems

In the course of our analyses, we encountered several mistakes and shortcomings that led to a cryptographic or security-relevant implementation being compromised. More precisely, we identified the following reasons for most of the security problems we came across:

No cryptographic protection In case of the UHF RFID transponder of Chap. 6, all data was exchanged without encryption or authentication. Combined with the fact that an adversary can significantly increase the communication ranges—a fact apparently not considered by the manufacturer—this leads to a number of potential risks, including the duplication of transponder from a distance and the detection of a specific RFID.

Proprietary cryptography As a positive outcome, most DUTs were built using peer-reviewed, standard primitives. Only the SimonsVoss electronic lock uses a proprietary and not publicly reviewed cryptographic function in its authentication protocol—a design decision that resulted in the highly efficient cryptanalytical attacks described in [SDK+13].

No SCA countermeasures For most of the analyzed DUTs, no countermeasures against SCA were present at all. Only the DESFire MF3ICD40 attempts to hide the side-channel leakage by using asynchronous circuits and randomizing the timing of the 3DES. We suspect these countermeasures to be the reason for the relatively high amount of 250,000 required traces. Note that for the SimonsVoss lock and the Altera Stratix II, the undisclosed cryptographic methods do not reliably prevent an SCA. Using reverse-engineering techniques, we were able to completely recover the scheme implemented by the vendor and subsequently mount a side-channel attack.

Single Point of Failure (SPOF) All considered DUTs are based on symmetric key cryptography. Apparently, public key cryptography is only slowly being adopted for embedded applications and is as of today mostly present in high-security devices, e.g., electronic passports [BSI10]. For the DUTs of this thesis, the symmetric key can become a SPOF if no suitable key derivation scheme is in place. For instance, the Opencard analyzed in Sect. 7.5 uses keys (for certain applications) that are identical for all Opencards. Thus, having obtained these keys once, an adversary can compromise the respective application for any card in the field. The existence of a SPOF is particularly problematic in the case of the SimonsVoss lock. Even though a key derivation mechanism is used, an adversary can extract the system-wide master key from the unprotected hardware of a lock with SCA, rendering a complete installation insecure.

Table 12.2 gives an overview of the security problems applying to each DUT. For several DUTs, the existence of a SPOF depends on the way the surrounding system is configured, marked with a ⓘ in Tab. 12.2. This is the case if a suitable key derivation scheme could avoid giving all devices in an installation an identical key.

For the SimonsVoss system, storing a master key on each lock cannot be avoided: the lock has to *verify* the authenticity of a transponder and hence to be able to derive the key of any

DUT	No crypto	Proprietary crypto	No SCA counter-measures	SPOF
UHF RFID	×	n/a	n/a	n/a
DESFire MF3ICD40				ⓘ
SimonsVoss Lock		×	×	×
Yubikey 2			×	ⓘ
Maxim SHA-1 DS28E01			×	ⓘ
Maxim SHA-1 DS2432			×	ⓘ
Altera Stratix II FPGA			×	×

Table 12.2: Security problems of the DUTs of Chap. 6–11. × indicates that a specific security problem applies, ⓘ that the problem may apply depending on the system design

transponder in the system. In contrast, most other DUTs in this thesis *prove* their authenticity so each device could get an individual key. We further discuss suitable key diversification methods in Sect. 12.5.3. Still, it is at least questionable whether, e.g., the manufacturer of an ink cartridge using a Maxim SHA-1 IC implements a suitable scheme in practice or simply uses a global, fixed key for one specific product line.

Finally, note that for protecting the bitstream of an FPGA in general and for Stratix II in particular, diversifying the decryption key cannot prevent an adversary from recovering the bitstream containing the vendor's IP. A single successful attack on one FPGA used on a specific product enables the decryption of the bitstream. Hence, for IP protection of FPGAs, suitable countermeasures on the physical level as described in Sect. 12.5.1 are of particular importance.

12.4 Responsible Disclosure

As mentioned in Sect. 1.1, we informed the vendor of a specific product whenever we had found a practical vulnerability several months before publishing our results. Following the principles of responsible disclosure, we left out details in certain cases, e.g., for the SimonsVoss electronic lock. Depending on the particular situation, we discussed our findings and possible methods to protect the product with the manufacturer. Often, changes on the protocol (cf. Sect. 12.5.2) or backend level (cf. Sect. 12.5.3) could significantly reduce the threat posed by our attacks, without requiring traditional countermeasures (cf. Sect. 12.5.1).

12.5 Countermeasures

To prevent or at least mitigate the implementation attacks presented in this thesis, a multitude of countermeasures can be applied. Here, we distinguish three levels on which a cryptographic device or the whole surrounding system can be protected: *(i)* by employing countermeasures for the actual hardware and the algorithms (Sect. 12.5.1), *(ii)* by making certain design choices for the used protocols (cf. Sect. 12.5.2), and *(iii)* by designing the backed in a suitable manner (Sect. 12.5.3).

12.5.1 Physical and Algorithmic Level

Makeshift solutions for protecting an embedded device against SCA tend to focus on removing the source of the leakage only. Accordingly, many engineers—when initially confronted with an SCA problem—propose to balance the power consumption of a device to avoid the dependency of a trace on the processed value. However, in our opinion, there are several important points to consider that are beyond such "simple" countermeasures when it comes to preventing implementation attacks.

Secure Hardware

Especially for high-security devices, the use of hardware specifically designed to withstand physical attacks is mandatory. Apart from detection circuits, e.g., light sensors, clock filters, and supply voltage monitoring, for various FI techniques, a strong protection against reverse-engineering is of special importance in our eyes.

The example of the SimonsVoss electronic lock in Chap. 8 demonstrates the consequences that can arise when security-relevant algorithms with system-wide keys are implemented on unprotected hardware. Having overcome the (weak) read-out protection of the PIC μC once, the full embedded code is in the hands of the adversary, enabling both cryptanalytical attacks [SDK+13] and SCA—which would have been impossible or at least extremely time-consuming without the knowledge of the implementation.

Suitable countermeasures to protect the IP, i.e., the embedded code, an FPGA bitstream, or functions directly implemented in silicon, include the shielding of the configuration bits, the use of (active) meshes to prevent microprobing, "obfuscating" the layout of the IC to make reverse-engineering difficult, and using a modern technology with a small feature size. On a higher level, the data bus can be encrypted so that an adversary cannot directly interpret data obtained by, e.g., placing microprobes on the respective wires.

Of course, all these techniques can be circumvented, as, e.g., shown by the example of the Infineon SLE 66PE that was successfully attacked by Tarnovsky [Tar10] using a FIB. Data bus encryption schemes can be reverse-engineered using microscope photos of a chip die [NT11]. Still, the presence of according protection mechanism significantly raises the bar for an adversary. Thus, employing an appropriate hardware platform, e.g. a modern smartcard or a secure μC is highly recommendable.

Countermeasures on the Circuit Level

Various methods have been described in the literature and been applied in practice for preventing side-channel attacks by designing the underlying IC in a specific way. Custom logic styles that attempt to balance the power consumption and thus eliminate the leakage have been extensively studied, cf., e.g., [TAV02, TV03, PM05, MOP07]. A major problem with this approach is that usually, complementary wires with as identical characteristics as possible have to be realized. In practice, due to manufacturing tolerances, this turns out to be difficult. Hence, in many cases, a residual leakage remains and SCA can still be applied (but may require a larger number of traces to succeed).

The opposite approach to balancing is the generation of noise, e.g., by randomly switching a load while the cryptographic operation is executed. Both countermeasures can be regarded as

"hiding" in the amplitude dimension [MOP07]. While balancing reduces the SNR by decreasing the amplitude of the leakage signal, additional noise achieves the same by increasing the noise level.

The second dimension available for hiding side-channel information is the time. For example, the use of asynchronous circuits, for which the signal propagation time depends on environmental conditions, has been proposed to randomize the execution timing of cryptographic functions [FML+03]. In a similar way, the clock signal of a synchronous circuit can be frequency-modulated to remove the time synchronization between side-channel traces.

Again, methods exist to bypass most of these countermeasures. For instance, in the case of the Mifare DESFire MF3IC40, we have reason to believe that several circuit-level protection mechanisms against SCA are in place. Still, as demonstrated in Chap. 7, special processing techniques and a, compared to the other DUTs, higher number of traces enabled us to successfully extract the full 3DES key. In general, as shown in Sect. 2.5, DSP techniques can significantly reduce the influence of timing variations and additional noise on the success rate of SCA. Nevertheless, countermeasures on the circuit level help to increase the difficulty of side-channel attacks and prevent straightforward "schoolbook" SCA.

Countermeasures on the Algorithmic Level

While SCA protection on the circuit level is largely independent of the specific cryptographic scheme, countermeasures on an algorithmic level require the designer to take the properties of the implemented primitive, e.g., AES or DES, into account. However, certain general classes of techniques can still be identified.

Similar to changing the clock frequency mentioned in Sect. 12.5.1, side-channel traces can be de-synchronized by randomizing the order and the timing of the steps performed in a cryptographic algorithm [MOP07]. Depending on the type of the implementation, the granularity can vary. For example, a hardware realization of the AES may randomly insert "dummy" rounds that do not affect the output. In a software implementation, dummy instructions could be inserted within a single round, or the program execution could be randomly stopped by interrupts.

Besides, certain parts of an algorithm, e.g., the S-box look-ups in a symmetric cipher, are often identically and independently performed on small parts of the algorithm's state. Hence, the order of these operations can be changed at random, a technique referred to as shuffling, cf. [VCMKS12]. Again, the timing relationship between traces is disturbed, because in different traces, a particular S-box look-up occurs at different points in time.

Masking schemes [CJRR99, CG00, RP10] aim at removing the dependency between a predicted intermediate and the observed side-channel leakage by adding a mask—a value that differs in each execution—to the state of the algorithm. Since the mask is generated internally and never output by the device, an adversary cannot compute valid hypotheses in differential attacks like DPA or CPA. Devising suitable masking schemes has become a separate research sub-area and a wide variety of schemes with different security properties and resource requirements is available. Nevertheless, adding masking to a cryptographic circuit generally results in a significant (time and/or space) overhead and thus can be prohibitive for constrained device, e.g., a limited 8-bit µC as it is employed on the Yubikey 2 (Chap. 9) or the SimonsVoss lock (Chap. 8).

In general, numerous other proposals for protecting different cryptographic schemes can be found in the literature. For asymmetric schemes like RSA and ECC, the secret exponent or scalar can be masked by adding a random multiple of the group order [MOP07]. The authors of [STA+10] propose a countermeasure using infective computations to protect such schemes against CIA, i.e., a combination of SCA and FI, an attack vector initially introduced in [AVFM07].

Impeding Reverse-Engineering

As already mentioned in Sect. 12.5.1, protecting the embedded code of a µC or the bitstream of an FPGA is a basic requirement for building a secure system. If the design IP is in the hands of an adversary, he is not only able to duplicate the respective product, but can also perform a detailed low-level analysis of the security features. For example, the attacks on the SimonsVoss system in Chap. 8 were enabled by the prior reverse-engineering of the code running on the electronic lock and the transponder. Similarly, the SCA of the Stratix II FPGAs was made possible by extracting the proprietary encryption scheme from the Quartus II PC software, cf. Chap. 11.

If hardware countermeasures against the extraction of IP cannot be employed or if program code is implicitly in the hand of an adversary (as it is the case for the Quartus II PC software), the reverse-engineering process can be made more difficult by using obfuscation techniques [CT02]. A theoretical discussion of obfuscation was published in [BGI+01], whereas numerous heuristic approaches have been in use for a long time.

Even though obfuscation techniques cannot completely exclude the possibility of reverse-engineering, they make the process far more time consuming. Considering the cases described in this thesis, the analysis of the Quartus II software turned out to be relatively straightforward since the shipped binary included debug symbols, i.e., the original function names assigned by the developers. This fact allowed us to directly pinpoint and analyze the relevant functions. For the less complex embedded code running on the PIC µC of the SimonsVoss lock, a considerably longer time was necessary to understand the relevant functionality. Although no deliberate obfuscation seems to have been used, the machine code was likely generated by an optimizing compiler for a high-level programming language like `C`. Thus, additional efforts were required to identify and reverse-engineer the relevant cryptographic functions.

Unlike µCs and PC software, the configuration bitstream of FPGAs was long thought to be extremely difficult to interpret, although the threat of reverse-engineering has already been discussed in 2004 [WGP04]. Still, FPGA vendors deny this possibility or consider it as unlikely. For instance, Altera states in a document [alt]:

> "Reverse engineering any high-density FPGA design through the configuration file is very difficult and time-consuming, even without encryption, as the configuration file contains millions of bits. In addition, Altera's configuration file formats are proprietary and confidential."

This statement appears in a different light today, considering that open-source tools like `debit` [deb] are publicly available for creating a netlist from an FPGA bitstream. Several publications [NR08, BSH12] deal with different approaches for FPGA reverse-engineering. Thus,

it seems likely that further progress in this area will continue to relax the requirements in this regard.

In the context of SCA countermeasures, it should be noted that a successful reverse-engineering turns a black-box into a white-box attack. Subsequently, this allows an adversary to circumvent SCA countermeasures. For example, if masking is applied, the relevant points of the power trace for higher-order attacks [Mes00] could be determined by analyzing the concrete implementation. Besides, in [DRS+13] it is shown that the knowledge of the embedded code of an AES operation can be used to create a model that automatically removes dummy instructions inserted to protect the algorithm against SCA.

12.5.2 Protocol Level

On the level of protocol, i.e., the way cryptographic primitives are applied and combined, measures can be taken to prevent implementation attacks or to limit the consequences of a successful key extraction. First, it should be made sure that in all cases standard, peer-reviewed primitives like AES, 3DES, RSA, or ECC are employed in a secure manner. Proprietary cryptography (as, e.g., used by the SimonsVoss system) can, once the code has been extracted by means of a physical UV-C attack, be reverse-engineered and broken.

Even if secure primitives are used, applying them in a non-standard way can lead to unexpected problems. For instance, Altera implemented a key derivation function on the Stratix II to prevent the direct cloning of FPGA designs in case the bitstream encryption is broken (e.g., by means of SCA), cf. Sect. 11.2.3. However, instead of using a standard hash construction based on block ciphers [MvOV96, Chap. 9], the designers employed the AES in a manner that does not provide second-preimage resistance.

Regarding SCA-specific countermeasures, straightforward approaches, e.g., limiting the number of traces or reducing the rate at which measurements can be acquired, can be effective in certain situations. For example, for the Yubikey 2 (Chap. 9), some delay between the generation of two OTPs is acceptable for the user. This leads to a maximum acquisition rate of $11.1\,{}^{\text{traces}}/_{\text{min}}$ that cannot be reduced. If the attack would not have succeed with the relatively low amount of 666 EM traces, the practical threat would be considerably reduced due to the slow measurement process. In contrast, the attack on the DESFire MF3IC40 was enabled by having access to a large number of traces, which could be recorded at a rate of approximately $595.23\,{}^{\text{traces}}/_{\text{min}}$. If the DESFire MF3ICD40 would require a complete shutdown and re-initialization after a failed authentication (due to SCA measurements), the current time of approximately 7 h to obtain the required traces could have been significantly increased.

If non-volatile memory is available, a device could furthermore count the number of invocations of the cryptographic functionality, for instance, to throttle the execution time after a "suspicious" number of attempts or to allow for the detection of an attack in a subsequent forensic analysis.

Finally, additional measures like assigning and checking a fixed UID for each device are beneficial. Taking RFID smartcards as an example, an adversary who successfully extracted the secret key still has to employ a custom emulator, e.g., the Chameleon [KvMOP11], to create a functionally identical clone. This in turn increases the effort for large-scale attacks and the risk of the adversary being detected when using a suspicious-looking device instead of a genuine card.

12.5.3 System Level

Embedded devices are often integrated into a surrounding system: public transport cards interact with ticketing terminals, access control transponders communicate with corresponding readers, and the validity of an OTP generated by a token is checked on a central server. A suitable design of the overall system and especially the backend—which is usually not accessible to an adversary—can thus provide a sufficient security level even though secret keys may have fallen into the hands of an attacker. A comprehensive summary in the context of RFID-based access control systems is given in [RNP10].

Backend Checks

Whenever a database is present in a backend, e.g., to keep track of payments, accesses to a specific area, or login attempts with an OTP token, automated checks can be performed to detect and report suspicious or irregular activities. For instance, if an RFID smartcard is registered at several locations within a short time frame, this may indicate that a duplicate of the card is in use. Logging which transponder was used to open an electronic lock may help in an investigation after an unauthorized access attempt. However, such methods could also severely affect a user's privacy and must thus be employed with care.

Blacklisting of devices possibly related to an (attempted) attack can, once the suspicious activity has been detected, prevent further damage, e.g., financial losses in case of a payment system or unauthorized access for electronic door locks.

Clearly, these countermeasures require an *online* system, in which components are permanently or at least in regular intervals connected to a central server. In a public transport system, the terminals installed in buses and subways could synchronize a local database with the backend when returning to the depot at night. In the context of access control, online systems can be realized as well. For example, SimonsVoss offers the so-called WaveNet extension for their electronic locks, which allows a central authority to re-configure the access rights, download log files, and so on.

In other cases, an online system is too costly or cannot be realized at all due to the scenario the device is used in. Connecting an FPGA to a background network to detect duplicated IP is impossible in automotive or military applications with high reliability requirements. Also, non-technical restrictions may apply. E.g., sending the UID of each ink cartridge (protected with a Maxim SHA-1) installed in a printer to the manufacturer likely violates laws that protect the individual's privacy or the right to use second-source products.

Key Diversification

Using a secure scheme for key diversification, i.e., ensuring that each device has individual, unique cryptographic keys, is almost mandatory for modern systems and not limited to embedded applications. It should always be taken into account that keys may be obtained by an adversary, whereas implementation attacks are only one of many possible ways for performing the key extraction.

Thus, to avoid a SPOF and to make sure that the compromise of one key does not render the whole system insecure, a global master key should be avoided or at least be stored in a particularly secure environment.

Basically, two solutions exist for diversifying keys: On the one hand, asymmetric cryptography, e.g., RSA or ECC, can be employed. Using a unique private/public key pair for each device, an adversary has to repeat the key extraction for each device to be compromised. Especially for SCA that, even though it can be realized at a relatively low cost, still requires a significant amount of equipment and knowledge, this prevents the attacks on a large scale.

However, the secure use of asymmetric cryptography implies the need for a Public Key Infrastructure (PKI), which creates additional overhead and which is often difficult to maintain. Moreover, in constrained embedded applications, the computational cost of asymmetric cryptography is even today often too high.

Thus, the majority of the devices in the field today are based on symmetric cryptography, as it is the case for all DUTs analyzed in this thesis. In this case, the use of system-wide master keys cannot be avoided completely. However, key diversification is applicable for devices that are handed out to users and thus are easily accessible for a potential adversary.

Again considering the example of a public transport system, this means that the master key is only stored on the reader terminals that are under the control of the system operator. This secret is then used to derive the key of individual cards, e.g., based on a card's UID and additional information.

Accordingly, for the case of the DESFire MF3ICD40 and the DESFire EV1, the vendor NXP has published guidelines for secure key diversification mechanisms [NXP06, NXP10a]. For the DESFire MF3ICD40, a system-wide master key $k_M = (k_{M,1}, k_{M,2})$—split into two 64-bit halves $k_{M,1}$ and $k_{M,2}$—is used to essentially encrypt the UID of a given card in CBC mode. More precisely, to derive the two 64-bit halves $(k_{C,1}, k_{C,2})$ of the 3DES key $k_C = (k_{C,1}, k_{C,2})$ for a card with a given 7-byte UID uid, the following scheme is proposed in [NXP06]:

$$\begin{aligned} k_{C,1} &= \text{3DES}_{k_M}\left(k_{M,1} \oplus (\text{0x88}\ uid)\right) \\ k_{C,2} &= \text{3DES}_{k_M}\left(k_{M,2} \oplus k_{C,1}\right) \end{aligned}$$

Note that the master key should be protected as good as possible. The example of the SimonsVoss lock in Chap. 8 demonstrates the consequences of storing a system-wide key on an unprotected hardware platform. A low-complexity side-channel attack allows to extract the master key from one arbitrary lock, allowing an adversary to derive the key for any transponder in the system and hence to impersonate any legitimate user gaining all his access rights.

Thus, for all devices possessing the master key, a combination of the countermeasures presented in Sect. 12.5.1–12.5.2 is necessary to achieve a sufficient level of security. Using modern smartcard hardware or secure µCs and countermeasures against implementation attacks like SCA is in our opinion a suitable way to minimize the risk when symmetric cryptography—and hence the use of master keys—cannot be avoided.

Chapter 13

Directions for Future Research

To conclude this thesis, in this chapter, various problems for further investigation are proposed. These ideas for future research are mostly based on the results and problems of the analyses carried out in this thesis. We identified three main areas of interest: First, current SCA techniques have certain shortcomings for real-world attacks, e.g., the problem of time synchronization of side-channel traces. Second, the combination of SCA and reverse-engineering should be investigated both from a constructive (countermeasure-oriented) and an adversary's point of view. Third, the security of the "next generation" of cryptographic devices, e.g., smartcards, RFIDs, FPGAs, and electronic locks, deserves further attention.

Contents of this Chapter

13.1 Improving SCA Techniques

Although the statistical foundations of SCA have been thoroughly investigated, cf. [SMY09, WOS12], there are numerous open questions of both practical and theoretical nature to be answered. Here, we outline problems that emerged while performing practical attacks—and for which in our opinion no general solution exists or only limited research is available.

13.1.1 Measurement Methods

Current SCA is mostly based on either power consumption or near-field EM measurements. This approach has two drawbacks: First, the acquired traces have one value per point in time, summarizing all activity that is performed on the DUT in this moment. Information in spatial dimensions (i.e., *where* the activity occurs) is usually not available. Recent works have shown that the spatial information can substantially improve SCA, e.g., by placing a small EM probe over one S-box [HMH+13] or by recording the photonic side-channel [SNK+12]. Hence, it seems valuable to further investigate new measurement methods. For instance, it would be interesting to compare photonic measurements to spatially resolved EM traces. Such EM measurements could be recorded using an array of small EM probes, with each probe being connected to a

separate digitizer so that many traces are acquired in parallel. Of course, this requires custom hardware—however, with the availability of low-cost FPGAs, ADCs, and processes to build small PCB antennas, we think creating such devices is feasible in an academic environment.

SCA often requires direct physical access to the DUT. Yet, there have been several reports that SCA can be performed from a distance. For instance, Oren and Shamir demonstrated in [OS07] that the power consumption of simple passive UHF RFIDs is modulated onto the reader's signal and can be reconstructed by observing the backscatter caused by the transponder. This in turn allows to perform SPA from a distance to recover a password used to permanently disable the transponder. In 2012, it was shown that the EM emanation of modern smartphones and tablets captured from a distance of up to 3 m can be used to mount side-channel attacks [KR12]. Thus, the achievable ranges for "remote" SCA of μCs, FPGAs, and related devices should be determined in general. For example, in the case of access control systems, a side-channel key recovery performed from the outside without opening the reader or electronic lock would pose a significant threat.

13.1.2 Processing of Side-Channel Traces

As already described in Chap. 2, the success rate of SCA depends on suitable pre-processing of the measured traces. The method of Sect. 2.5 to automatically derive optimal linear transforms offers several starting points for further research: First of all, the employed numerical optimization algorithm was used "out-of-the-box". We believe that an algorithm adapted to the specific requirements of our method may lead to better results and avoid the problem of overfitting. Besides, the proposed optimization criterion could be replaced, utilizing a different metric for the distinguishability of the correct key candidates. It would also be interesting to investigate whether an analytical solution for the present or a different suitable optimization criterion can be computed efficiently. In this regard, the applicability of statistical methods like CCA in a side-channel context deserves attention. Note that we focused on CPA only. However, distinguishers like MIA [GBTP08] have been shown to be superior in certain cases. Thus, finding a similar technique to compute the mutual information of transformed traces without re-executing the complete MIA is worth further research.

In general, linear transforms only cover a subset of the necessary pre-processing steps in many practical cases. A major problem is the re-synchronization of traces being misaligned due to timing-based countermeasures (Sect. 12.5.1) or other (unintended) effects like clock drift. Here, "classical" re-alignment methods like pattern matching and DFA, cf. Sect. 2.4.2, only mitigate the problem but cannot be considered a general solution. Newer proposals, e.g., the use of DTW [vWWB11] or PCA [BHvW12], attempt to provide more universal methods, which, however, do not perform significantly better than traditional techniques in our experience. Thus, compensating the misalignment of side-channel traces deserves further attention. Statistical methods so far undiscovered or not applied in the context of SCA may relax or remove the requirement for temporal synchronization of the traces altogether.

13.2 Implementation Attacks and Reverse-Engineering

Considering the example of the SimonsVoss system, one can conclude that the success of implementation attacks depends on the knowledge of *implementation details*—without having reverse-

engineered the exact functionality, performing the side-channel attacks of Sect. 8.4 would have been close to impossible. In general, the more information is available about a target device, the more likely it is that an adversary can successfully perform an attack. Thus, Kerckhoffs's principle does not hold for protection mechanisms against implementation attacks. While the cryptographic algorithms should be peer-reviewed standards like AES or 3DES, the exact nature of countermeasures does not necessarily have to be disclosed. In fact, vendors of modern high-security smartcards, e.g., NXP or Infineon, only provide a high-level description of the employed protection mechanisms. All details are kept confidential to increase the difficulty for an adversary attempting to conduct implementation attacks.

However, reverse-engineering techniques are constantly progressing, both for circumventing readout protections and directly analyzing circuits on the level of the silicon die. It has to be taken into account that undisclosed and secret countermeasures become public, as it was the case for the memory encryption scheme of a wide-spread smartcard [NT11]. Thus, future research could focus on the offensive aspect, i.e., developing new attacks involving implementation details, and suitable defenses, e.g., advanced obfuscation schemes or countermeasures that remain secure when being reverse-engineered.

13.2.1 Attack Techniques

The security gained through countermeasures against SCA and related techniques like FI significantly reduces in practice if the concrete realization is known. For example, a higher-order DPA is easier to apply when an adversary can directly infer the relevant points in time from the implementation, rather than exhaustively testing all possible combinations. The randomization of an algorithm's timing can be circumvented when the type of the countermeasure, e.g., the insertion of dummy rounds, shuffling, and the parameters, e.g., the number of dummy rounds, are known. A specific, protected implementation can thus be weaker when assuming a reverse-engineering adversary compared to a black-box attack. It would thus be interesting to determine the security level of modern smartcards and other high-security devices in this regard.

Apart from applying the result of reverse-engineering, the actual techniques for obtaining implementation details can be improved. For example, new methods like the SCA-based disassembler described in [EPW10] may render traditional methods like readout protections ineffective. Further improvements could result from combining this method with space-resolved side-channels like the photonic emission. Even if the effort for extracting the internal functionality of an IC is relatively high, an adversary has to perform the reverse-engineering only once. After that, with full access to the details of an implementation, less elaborate and complex attacks may be discovered.

13.2.2 Countermeasures

Having identified the potentially severe threat posed by the combination of reverse-engineering and implementation attacks, the need for suitable countermeasures is evident. One approach is to increase the security of the underlying hardware itself to prevent an adversary from gaining access to information (e.g., embedded code, memory contents, or circuit photos) that can be used for reverse-engineering. For example, data buses can be encrypted and the silicon die be covered with an active shield to substantially complicate microprobing attacks carried out while

the IC is active. A recent proposal [BCC+12] studied the use of three-dimensional circuits to protect security-relevant parts of an IC.

A small technology size, e.g., a 32 nm or 22 nm process, rules out optical microscopy for taking pictures of an IC, thus requiring considerably more expensive and complicated equipment in the adversary's lab. Still, analyzing such a circuit is possible using advanced imaging techniques like Scanning Electron Microscopy (SEM) or Transmission Electron Microscopy (TEM). For instance, Chipworks published an analysis [int] of Intel's 22 nm CPUs, which started shipping in the first quarter of 2013. Nevertheless, a small feature size likely excludes low-budget adversaries and hence contributes to building more secure hardware.

As a second countermeasure, the designer can take into account that an adversary may eventually obtain "raw" data like IC photographs or the embedded code of an µC. Then, the goal is to make the subsequent reverse-engineering process, i.e., understanding the implemented functionality, as hard and time-consuming as possible. As described in Sect. 12.5.1, the (cryptographic) algorithm itself can be obfuscated. In the case of ICs, separate functional units can be mixed, a technique that NXP allegedly already used for the DESFire MF3ICD40 analyzed in Chap. 7. Furthermore, the routing of connections between different gates can be partially randomized, increasing the time needed for inferring the netlist if this step has to be performed (partially) manually. This is to our knowledge the case for the open-source circuit analysis tool `degate` [deg]. For FPGAs and ICs, the subsequent step, i.e., understanding the actual logic behind the netlist, can be further complicated by including irrelevant "dummy" circuitry or using non-standard data encoding, cf. [MBCB+11] for a discussion.

In the case of software implementations, the classical techniques known from PC applications can be applied, cf. Sect. 12.5.1. Yet, as an additional constraint, embedded applications very often have substantial size and/or performance requirements. Hence, countermeasures based on inserting large blocks of inactive code or highly nested control flow can be prohibitive. To our knowledge, relatively little research has been done with respect to finding efficient and at the same time secure obfuscation techniques in the context of embedded systems. Hence, this aspect is an interesting area of research.

13.3 Practical Implementation Attacks

As the practical attacks of Chap. 6–11 have shown, a wide variety of (embedded) devices is susceptible to implementation attacks, ranging from relatively straightforward eavesdropping on (unprotected) communication to side-channel key recovery attacks. In our opinion, it is mandatory to continue with the analysis of other devices (possibly belonging to recent or upcoming product generations) in order to constantly improve the security of embedded systems, which become more and more prevalent in all kinds of applications.

13.3.1 Physical Layer of RF Devices

For HF and UHF RFIDs, the achievable passive eavesdropping and active communication ranges have been extensively analyzed, e.g., cf. Chap. 6. For RF devices operating at low frequencies of a few hundred kHz, less research has been done to our knowledge, although such devices are wide-spread in many security-sensitive applications: For example, the SimonsVoss electronic locking system operates at 25 kHz, RFIDs for vehicle immobilizers commonly use a frequency of

125 kHz, and medical devices like pacemakers are programmed over a 175 kHz link [HHBR+08]. Although we are aware of (unpublished) long-range relay attacks on car keys at 125 kHz, relatively little systematic research has been done with respect to extending the range of LF communication. This gap should be closed, analyzing both the maximum distance for eavesdropping and for actively powering and communicating with such devices.

In the context of UHF RFID as discussed in Sect. 6.4, the DASH7 initiative is currently extending the ISO 18000-7 standard [ISO09] to include cryptographic protection against attacks as presented in this thesis [DAS10]. However, the question arises how secure the cryptographic protocols (based on shared-key AES-128) are *(i)* from a cryptanalytical point of view and *(ii)* if countermeasures against implementation attacks have been considered. At least, in [DAS10], the use of secure key diversification is mentioned, thus presumably avoiding a SPOF resulting from using an identical key for many transponders. Nevertheless, a detailed investigation is necessary once the first real products are available.

13.3.2 Contactless Smartcards and RFIDs

Having demonstrated the fundamental susceptibility of the DESFire MF3ICD40 towards SCA in Chap. 7, there are several directions for further research to consider: First, the SCA could be improved in order to work with a smaller number of traces, for instance, employing different alignment methods. Apart from that, extensions of the proposed TA may allow to reduce the error rate or to utilize the templates generated with a profiling device to recover the unknown key of another DESFire MF3ICD40 card. It would also be relevant to constructively use the side-channel leakage to identify one particular device. Initial experiments in this regard indicated that for the DESFire MF3ICD40, the identifying features mostly originate from the RF frontend as already discussed in [DHBv09] for similar RFIDs. Yet, using the side-channel leakage could lead to physical layer identification less dependent on the measurement setup.

Apart from that, the developed techniques for SCA of RFIDs can be applied in order to attack other cryptographic RFIDs, possibly including (certified) high-security smartcards as—for instance—used for electronic identity documents. Because according devices, e.g., the Mifare DESFire EV1, do not directly modulate the reader's field with a clearly observable leakage signal (cf. Sect. 4.5), using other measurement methods could lead to successful attacks. For example, high-resolution EM measurements combined with techniques to remove the reader's signal could be utilized to obtain more "structured" traces as a starting point for further analysis.

13.3.3 Access Control Systems

As shown for the SimonsVoss system 3060 "G2" in Chap. 8, cryptanalytical and implementation attacks can enable an adversary to overcome modern access control systems and gain unauthorized access with relatively low effort once the functionality has been reverse-engineered. Furthermore, for the particular case of SimonsVoss, not all possible attack vectors have been examined. So far, the attacks focused on the authentication protocol for unlocking a door. The programming functionality for configuring the access rights and also (partially) updating the firmware of the lock was not further analyzed. Nevertheless, this interface could be target of further attacks, for example, an adversary specifying additional access rights for his transponder. The firmware update function could be, if no or insufficient authentication and encryption is used, used to "infect" the lock with malware that, e.g., logs all opening events or disables or

opens the lock at a specific point in time in a DoS scenario. If a similar programming protocol was implemented on the transponder, this malware could spread from lock to lock via the transponders of the legitimate users.

Besides, SimonsVoss offers an add-on module operating at 868 MHz for performing administrative tasks without programming each door lock separately. This RF interface termed "Wave Net" by the vendor [Simb] could be subject to similar attacks. However, in contrast to the "normal" 25 kHz programming link, the achievable communication ranges at 868 MHz can be expected to be in the order of tens to hundreds of meters, cf. Sect. 6.4. Hence, if an attack vector exists, it could be exploited from a distance, reducing the risk for the adversary being detected and potentially affecting many doors at the same time.

In general, a comprehensive survey of the current security level of electronic locks and access control systems would be worthwhile. According to our observations, the systems of numerous other vendors are either comparable or worse than the SimonsVoss products in terms of security. Amongst others, we found schemes solely relying on a UID check and solutions based on extremely weak proprietary cryptography—providing less protection than the "close to secure" SimonsVoss system.

13.3.4 FPGA Bitstream Encryption

For virtually all current FPGA vendors, it has been shown that the anti-counterfeit protection can be circumvented with implementation attacks. The bitstream encryption key of Altera Stratix II and most Xilinx devices can be extracted by means of SCA, cf. Chap 11 and [MBKP11, MKP12], respectively. Similarly, for Actel's Flash-based ProASIC3 FPGA, a side-channel attack was used to recover the key for accessing an undocumented programming interface, allowing an adversary to read the stored bitstream [SW12a].

It is interesting to see when and to which extent FPGA vendors will react to this threat. For instance, in the case of Altera, since the Stratix II family belongs to an older product generation, it was somewhat expected that SCA countermeasures have been ignored during the development. However, recent families like Stratix V or Arria II probably feature an only slightly different scheme for bitstream encryption. Therefore, analyzing the security of the more recent Altera FPGAs from an SCA point of view is interesting for future work. A similar analysis should be performed for newer Actel and Xilinx FPGAs, e.g., the Virtex 6 and Virtex 7 series.

With respect to developing suitable countermeasures for the particularly problematic case of protecting FPGA bitstreams, further research is definitely necessary. Since the FPGA is permanently in the hands of the adversary and since extracting the key of one single device allows to obtain the bitstream and hence the IP for a product, a level of protection similar to modern smartcards would be desirable, e.g., certified according to CC or similar evaluation standards. As an additional requirement, note that bitstreams for modern FPGAs can be relatively large: the smallest Virtex 7 FPGA 7VX330T has a bitstream size of approximately 13.26 MB, the largest 7V2000T 53.33 MB [Xil13]. Hence, the decryption engine on the FPGA has to be designed with throughput in mind to avoid long configuration times. Because countermeasures usually lead to a significant increase of the execution time and/or size of the circuitry, finding and implementing highly secure and at the same time efficient mechanisms for protecting FPGA bitstreams is an interesting problem.

Part V

Appendix

Bibliography

[AARR03] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM Side-Channel(s). In *Cryptographic Hardware and Embedded Systems – CHES'02*, LNCS, pages 29–45. Springer, 2003.

[AIS08] AIST. *Side-channel Attack Standard Evaluation Board SASEBO-B Specification*, 2008. `http://www.risec.aist.go.jp/project/sasebo/download/SASEBO-B_Spec_Ver1.0_English.pdf`.

[alt] Protecting Intellectual Property Through FPGA Design Security. Advertorial. `http://www.altera.com/literature/ads/fpgadesignsecurity.pdf`.

[AN309] AN 341: Using the Design Security Feature in Stratix II and Stratix II GX Devices. Technical report, Altera, 2009. `http://www.altera.com/literature/an/an341.pdf`.

[Ana04] Analog Devices, Inc. *AD8045 Voltage Feedback High Speed Amplifier Datasheet*, 2004. `http://www.analog.com/static/imported-files/data_sheets/AD8045.pdf`.

[Ana09a] Analog Devices, Inc. *AD8058 Dual, High Performance Voltage Feedback 325 MHz Amplifier Datasheet*, 2009. `http://www.analog.com/static/imported-files/data_sheets/AD8057_8058.pdf`.

[Ana09b] Analog Devices, Inc. *AD9283: 8-Bit, 50 MSPS/80 MSPS/100 MSPS ADC Datasheet*, 2009. `http://www.analog.com/static/imported-files/data_sheets/AD9283.pdf`.

[Ana09c] Analog Devices, Inc. *AD9708 8-Bit, 100 MSPS+ TxDAC D/A Converter Datasheet*, 2009. `http://www.analog.com/static/imported-files/data_sheets/AD9708.pdf`.

[Atm] Atmel. *XMEGA Manual*, Version A edition.

[Atm03] Atmel. *ATmega163 Datasheet*, 2003. `http://www.atmel.com/dyn/resources/prod_documents/doc1142.pdf`.

[AV-13] AV-Test. Malware statistics. Report, 2013. `http://www.av-test.org/statistiken/malware/`.

[AVFM07] Frederic Amiel, Karine Villegas, Benoit Feix, and Louis Marcel. Passive and Active Combined Attacks: Combining Fault Attacks and Side Channel Analysis. In *Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC '07*, pages 92–102, Washington, DC, USA, 2007. IEEE Computer Society.

[Bal07] Baltech. *ID-engine Universal Contactless Smartcard and RFID Read Write Devices Datasheet*, 2007. `http://www.baltech.de/index.php?option=com_docman&task=doc_download&gid=56`.

[BCC+12] Sébastien Briais, Stéphane Caron, Jean-Michel Cioranesco, Jean-Luc Danger, Sylvain Guilley, Jacques-Henri Jourdan, Arthur Milchior, David Naccache, and Thibault Porteboeuf. 3D Hardware Canaries. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES'12*, volume 7428 of *LNCS*, pages 1–22. Springer, 2012.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In *Advances in Cryptology – CRYPT0'96*, LNCS, pages 1–15, London, UK, UK, 1996. Springer.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES'04*, volume 3156 of *LNCS*, pages 16–29. Springer, 2004.

[BDG+10] Michael Backes, Markus Dürmuth, Sebastian Gerling, Manfred Pinkal, and Caroline Sporleder. Acoustic side-channel attacks on printers. In *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, pages 20–20, Berkeley, CA, USA, 2010. USENIX Association.

[BDH+97] F. Bao, R. H. Deng, Y. Han, A.Jeng, A. D. Narasimhalu, and T. Ngair. Breaking Public Key Cryptosystems on Tamper Resistant Devices in the Presence of Transient Faults. In *Proceedings of the 5th International Workshop on Security Protocols*, pages 115–124. Springer, 1997.

[BDL97] Dan Boneh, Richard A. Demillo, and Richard J. Lipton. On the Importance of Checking Computations. In *Proceedings of Eurocrypt'97*, pages 37 – 51, 1997.

[Bec88] Friedrich Beck. *Präparationstechniken für die Fehleranalyse an integrierten Halbleiterschaltungen*. VCH Verlagsgesellschaft, 1988.

[Ber] Daniel J Bernstein. Cache-timing attacks on AES. `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf`.

[BFK+12] Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, and Joe-Kai Tsay. Efficient padding oracle attacks on cryptographic hardware. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO'12*, volume 7417 of *LNCS*, pages 608–625. Springer, 2012.

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO'01*, volume 2139 of *LNCS*, pages 1–18. Springer, 2001.

[BGS+05] Stephen C. Bono, Matthew Green, Adam Stubblefield, Ari Juels, Aviel D. Rubin, and Michael Szydlo. Security analysis of a cryptographically-enabled RFID device. In *Proceedings of the 14th conference on USENIX Security Symposium*, volume 14. USENIX Association, 2005. http://www.usenix.org/events/sec05/tech/bono/bono.pdf.

[BHvW12] Lejla Batina, Jip Hogenboom, and Jasper van Woudenberg. Getting More from PCA: First Results of Using Principal Component Analysis for Extensive Power Analysis. In Orr Dunkelman, editor, *Topics in Cryptology – CT-RSA'12*, volume 7178 of *LNCS*, pages 383–397. Springer, 2012.

[Bla03] Matt Blaze. Rights Amplification in Master-Keyed Mechanical Locks. *IEEE Security & Privacy*, 1(2):24–32, 2003. http://www.crypto.com/papers/mk.pdf.

[Bog07] Andrey Bogdanov. Improved Side-Channel Collision Attacks on AES. In Carlisle Adams, Ali Miri, and Michael Wiener, editors, *Selected Areas in Cryptography – SAC'07*, volume 4876 of *LNCS*, pages 84–95. Springer, 2007.

[BOS09] Joppe W. Bos, Dag Arne Osvik, and Deian Stefan. Fast Implementations of AES on Various Platforms. *IACR Cryptology ePrint Archive*, 2009:501, 2009. http://eprint.iacr.org/2009/501.

[BPT10] Alessandro Barenghi, Gerardo Pelosi, and Yannick Teglia. Improving first order differential power attacks through digital signal processing. In *ACM-SIGSAC International Conference on Security of Information and Networks*, pages 124–133. ACM, 2010.

[BPT11] Alessandro Barenghi, Gerardo Pelosi, and Yannick Teglia. Information leakage discovery techniques to enhance secure chip design. In Claudio Agostino Ardagna and Jianying Zhou, editors, *WISTP*, volume 6633 of *LNCS*, pages 128–143. Springer, 2011.

[Bra10] Christian Brandt. Hacking iButtons. Presentation at 27C3, 2010. http://cribert.freeforge.net/27c3/ibsec.pdf.

[Bre11] Tom Brewster. Does cyber crime really cost £27 billion a year? Article on ITPro, 2011. http://www.itpro.co.uk/631189/does-cyber-crime-really-cost-27-billion-a-year.

[Bri10] Brightsight. Unique Tools from the Security Lab. Website as of June 1, 2010, 2010. http://www.brightsight.com/tools/sideways.

[Bri11] Peter Bright. RSA finally comes clean: SecurID is compromised, June 2011. http://arstechnica.com/security/2011/06/rsa-finally-comes-clean-securid-is-compromised/.

[BS97] Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology – CRYPTO'97*, pages 513 – 525, 1997.

[BSH12] F. Benz, A. Seffrin, and S.A. Huss. Bil: A tool-chain for bitstream reverse-engineering. In *22nd International Conference on Field Programmable Logic and Applications – FFL'12*, pages 735–738, 2012.

[BSI] BSI – German Ministry of Security. Security mechanisms in electronic ID documents. `http://www.bsi.de/fachthem/epass/`.

[BSI08] BSI – German Ministry of Security. Mifare DESFire8 MF3ICD81 Public Evaluation Documentation. Website as of October 2008, October 2008.

[BSI10] BSI – German Ministry of Security. Technical Guideline TR–03110 Advanced Security Mechanisms for Machine Readable Travel Documents. Website as of October 2010, October 2010. `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/TR-03110_v205_pdf.pdf?__blob=publicationFile`.

[But13] Eric Butler. FareBot Android App. Website, May 2013. `https://play.google.com/store/apps/details?id=com.codebutler.farebot&hl=en`.

[Car05] Dario Carluccio. Electromagnetic Side Channel Analysis for Embedded Crypto Devices. Master's thesis, Ruhr-University Bochum, 2005.

[CCD00] Christophe Clavier, Jean-Sèbastien Coron, and Nora Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In Çetin Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'00*, volume 1965 of *LNCS*, pages 13–48. Springer, 2000.

[CFG+10] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal Correlation Analysis on Exponentiation. In Miguel Soriano, Sihan Qing, and Javier López, editors, *Information and Communications Security – ICICS'10*, volume 6476 of *LNCS*, pages 46–61. Springer, 2010.

[CG00] Jean-Sébastien Coron and Louis Goubin. On Boolean and Arithmetic Masking against Differential Power Analysis. In Çetin K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'00*, volume 1965 of *LNCS*, pages 231–237. Springer, 2000.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In Michael Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *LNCS*, pages 398–412. Springer, 1999.

[COM] COMELEC department, Tèlècom ParisTech. DPA Contest v2. Website as of February 2012. `http://www.dpacontest.org/v2/index.php`.

[Cou09] Nicolas Courtois. The Dark Side of Security by Obscurity - and Cloning MiFare Classic Rail and Building Passes, Anywhere, Anytime. In *SECRYPT*, pages 331–338. INSTICC, 2009.

[CRC] On-line CRC calculation and free library. http://www.lammertbies.nl/comm/info/crc-calculation.html.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'02*, volume 2523 of *LNCS*, pages 13–28. Springer, 2002.

[CT02] Christian S. Collberg and Clark Thomborson. Watermarking, tamper-proofing, and obfuscation: Tools for software protection. *IEEE Trans. Softw. Eng.*, 28(8):735–746, August 2002.

[Cur12] Sam Curry. Don't Believe Everything You Read... Your RSA SecurID Token is Not Cracked. blog entry, June 2012. http://blogs.rsa.com/dont-believe-everything-you-read-your-rsa-securid-token-is-not-cracked/.

[DAS10] DASH7 Alliance Security Working Group. DASH7 Security and Privacy, Part 1: Introduction. Whitepaper, 2010. www.dash7.org/whitepapers.

[deb] debit – Reverse-engineering tools for FPGA bitstreams. Website as of May 23, 2013. https://code.google.com/p/debit/.

[deg] degate – VLSI reverse engineering of digital logic in integrated circuits. Website as of June 1, 2013. http://www.degate.org/.

[Det11] Detica Limited. The Cost of Cybercrime. Report in partnership with the office of cyber security and information, 2011. https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/60942/THE-COST-OF-CYBER-CRIME-SUMMARY-FINAL.pdf.

[Dev09] Nicolas Devillard. iniParser: stand-alone ini Parser library in ANSI C. Website, October 2009. http://ndevilla.free.fr/iniparser/.

[DHBv09] Boris Danev, Thomas S. Heydt-Benjamin, and Srdjan Čapkun. Physical-layer identification of RFID devices. In *Proceedings of the USENIX security symposium – SSYM'09*, pages 199–214, Berkeley, CA, USA, 2009. USENIX Association.

[DRS+13] François Durvaux, Mathieu Renauld, François-Xavier Standaert, Loic Oldeneel tot Oldenzeel, and Nicolas Veyrat-Charvillon. Efficient Removal of Random Delays from Embedded Software Implementations Using Hidden Markov Models. In Stefan Mangard, editor, *Smart Card Research and Advanced Applications – CARDIS'12*, volume 7771 of *LNCS*, pages 123–140. Springer, 2013.

[EG12] M. Abdelaziz Elaabid and Sylvain Guilley. Portability of templates. *Journal of Cryptographic Engineering*, 2(1):63–74, 2012.

[EKM+08] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme. In

Advances in Cryptology – CRYPTO'08, volume 5157 of *LNCS*, pages 203–220. Springer, 2008.

[EPW10] Thomas Eisenbarth, Christof Paar, and Björn Weghenkel. Building a Side Channel Based Disassembler. *Transactions on Computational Science X - Special Issue on Security in Computing, Part I*, 6340:78–99, 2010.

[Ett13a] Ettus Research. Daughterboards. Website, April 2013. `https://www.ettus.com/product/category/Daughterboards`.

[Ett13b] Ettus Research. Homepage. Website, April 2013. `http://www.ettus.com/home/`.

[FH08] J. Ferrigno and M. Hlavac. When AES blinks: introducing optical side channel. *Information Security, IET*, 2(3):94–98, 2008.

[Fin03] Klaus Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley and Sons, 2nd edition, 2003.

[FK] Thomas Finke and Harald Kelter. Radio Frequency Identification - Abhörmöglichkeiten der Kommunikation zwischen Lesegerät und Transponder am Beispiel eines ISO14443-Systems.

[FML+03] Jacques J. A. Fournier, Simon Moore, Huiyun Li, Robert Mullins, and George Taylor. Security Evaluation of Asynchronous Circuits. pages 137–151, 2003.

[Fri46] T. H. Friis. A Note on a Simple Transmission Formula. In *Proceedings of the IRE*, number 34, 1946.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual Information Analysis – A Generic Side-Channel Distinguisher. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES'08*, volume 5154 of *LNCS*, pages 426–442. Springer, 2008.

[GHT05] Catherine Gebotys, Simon Ho, and C. Tiu. EM Analysis of Rijndael and ECC on a Wireless Java–Based PDA. In Josyula Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES'05*, volume 3659 of *LNCS*, pages 250–264. Springer, 2005.

[GKGM+08] Flavio D. Garcia, Gerhard Koning Gans, Ruben Muijrers, Peter Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. Dismantling MIFARE Classic. In *Proceedings of the 13th European Symposium on Research in Computer Security – ESORICS 2008*, LNCS, pages 97–114. Springer, 2008.

[GKGVM12] Flavio D. Garcia, Gerhard Koning Gans, Roel Verdult, and Milosch Meriac. Dismantling iClass and iClass Elite. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *Computer Security – ESORICS'12*, volume 7459 of *LNCS*, pages 697–715. Springer Berlin Heidelberg, 2012.

[GNU13] GNURadio project. GNURadio main page. Website, April 2013. `http://gnuradio.org/redmine/`.

[Gol08] Martin Goldack. Side-Channel based Reverse Engineering for Microcontrollers. Diploma thesis, Ruhr-University Bochum, 2008. https://www.emsec.rub.de/media/attachments/files/2012/10/da_goldack.pdf.

[Gra00] Joe Grand. Attacks on and Countermeasures for USB Hardware Token Devices. In *Proceedings of the Fifth Nordic Workshop on Secure IT Systems Encouraging Cooperation*, pages 35–57, 2000.

[Gra04] Joe Grand. Hardware Token Compromises. Presentation at Black Hat USA 2004, 2004. http://www.blackhat.com/presentations/bh-usa-04/bh-us-04-grand/grand_hardware_token_US04.pdf.

[Han05] Gerhard Hancke. A practical relay attack on ISO 14443 proximity cards. http://www.cl.cam.ac.uk/~gh275/relay.pdf, 2005.

[Han06] G. P. Hancke. Practical Attacks on Proximity Identification Systems (Short Paper). In *Proceedings of SP'06*, pages 328–333. IEEE Computer Society, 2006.

[Han08] G.P. Hancke. Eavesdropping Attacks on High-Frequency RFID Tokens. In *Workshop on RFID Security – RFIDSec'08*, 2008.

[HCN+06] Hagai Bar-El Hamid, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The Sorcerer's Apprentice Guide to Fault Attacks. In *Proceedings of the IEEE*, volume 94, 2006.

[HH11] Ludger Hemme and Lars Hoffmann. Differential Fault Analysis on the SHA1 Compression Function. In *Proceedings of the 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC'11*, LNCS, pages 54–62, Washington, DC, USA, 2011. IEEE Computer Society.

[HHBR+08] D. Halperin, T.S. Heydt-Benjamin, B. Ransford, S.S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W.H. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *IEEE Symposium on Security and Privacy – SP'08*, pages 129–142, 2008.

[HMF07] Michael Hutter, Stefan Mangard, and Martin Feldhofer. Power and EM Attacks on Passive 13.56 MHz RFID Devices. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES'07*, volume 4727 of *LNCS*, pages 320–330. Springer, 2007.

[HMH+13] Johann Heyszl, Dominik Merli, Benedikt Heinz, Fabrizio De Santis, and Georg Sigl. Strengths and Limitations of High-Resolution Electromagnetic Field Measurements for Side-Channel Analysis. In Stefan Mangard, editor, *Smart Card Research and Advanced Applications – CARDIS'12*, volume 7771 of *LNCS*, pages 248–262. Springer, 2013.

[HSST03] David R. Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical Correlation Analysis; An Overview with Application to Learning Methods. http://eprints.ecs.soton.ac.uk/9225/, May 2003.

[Hua05] Andrew Huang. Hacking the PIC 18F1320, 2005. available at http://www.bunniestudios.com/blog/?page_id=40, as of October 7, 2013.

[IDA] IDA Disassembler and Debugger. Website as of April 2013. https://www.hex-rays.com/products/ida/index.shtml.

[iee09] This Car Runs on Code. Article in IEEE Spectrum, February 2009. http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code.

[int] Intel's 22nm Tri-gate Transistors Exposed. Website as of May 23, 2013. http://www.chipworks.com/blog/technologyblog/2012/04/23/intels-22-nm-tri-gate-transistors-exposed/.

[Int01a] International Organization for Standardization. ISO/IEC 14443-3: Identification Cards – Contactless Integrated Circuit(s) Cards – Proximity Cards – Part 2: Radio frequency power and signal interface, February 2001.

[Int01b] International Organization for Standardization. ISO/IEC 14443-3: Identification Cards – Contactless Integrated Circuit(s) Cards – Proximity Cards – Part 3: Initialization and Anticollision, February 2001.

[Int01c] International Organization for Standardization. ISO/IEC 14443-4: Identification cards – Contactless Integrated Circuit(s) Cards – Proximity Cards – Part 4: Transmission Protocol, February 2001.

[Int09] International Organization for Standardization. ISO/IEC 15693-3: Identification Cards – Contactless Integrated Circuit Cards – Vicinity Cards – Part 3: Anticollision and Transmission Protocol, April 2009.

[Int11] International Telecommunication Union. Internet users per 100 inhabitants, 2001-2011. ITU World Telecommunication / ICT Indicators database, 2011. http://www.av-test.org/statistiken/malware/.

[ISO09] ISO/IEC 18000-7. Radio frequency identification for item management – Part 7: Parameters for active air interface communications at 433 MHz, 2009. www.iso.org.

[JMW05] A. Juels, D. Molnar, and D. Wagner. Security and Privacy Issues in E-Passports. In *Proceedings of SecureComm'05*, pages 74–88. IEEE Computer Society, September 2005.

[Jon12] Tom Jones. Prague's Opencard payment system now 'cardless'. Website as of February 2012, February 2012. http://www.ceskapozice.cz/en/business/companies/prague%E2%80%99s-opencard-payment-system-now-%E2%80%98cardless%E2%80%99.

[KCP07] Timo Kasper, Dario Carluccio, and Christof Paar. An Embedded System for Practical Security Analysis of Contactless Smartcards. In *WISTP*, volume 4462 of *LNCS*, pages 150–160. Springer, 2007.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Advances in Cryptology – CRYPTO'99*, LNCS, pages 388–397. Springer, 1999.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.

[KOP09] Timo Kasper, David Oswald, and Christof Paar. EM Side–Channel Attacks on Commercial Contactless Smartcards Using Low-Cost Equipment. In Heung Youl Youm and Moti Yung, editors, *10th International Workshop on Information Security Applications – WISA'09*, LNCS, pages 79–93. Springer, 2009.

[KOP10] Timo Kasper, David Oswald, and Christof Paar. A Versatile Framework for Implementation Attacks on Cryptographic RFIDs and Embedded Devices. In Marina Gavrilova, C. Tan, and Edward Moreno, editors, *Transactions on Computational Science X*, volume 6340 of *LNCS*, pages 100–130. Springer, 2010.

[KOP11a] T. Kasper, D. Oswald, and C. Paar. Wireless security threats: Eavesdropping and detecting of active RFIDs and remote controls in the wild. In *19th International Conference on Software, Telecommunications and Computer Networks – SoftCOM'11*, pages 1–6, 2011.

[KOP11b] Timo Kasper, David Oswald, and Christof Paar. Security of Wireless Embedded Devices in the Real World. In Norbert Pohlmann, Helmut Reimer, and Wolfgang Schneider, editors, *Information Security Solutions (ISSE) 2011 – Securing Electronic Business Processes*, pages 174–189. Vieweg+Teubner, 2011. `http://www.viewegteubner.de/Buch/978-3-8348-1911-6/ISSE-2011-Securing-Electronic-Business-Processes.html`.

[KOP12] Timo Kasper, David Oswald, and Christof Paar. Side-Channel Analysis of Cryptographic RFIDs with Analog Demodulation. In Ari Juels and Christof Paar, editors, *7th International Workshop on RFID Security and Privacy – RFIDSec'11*, volume 7055 of *LNCS*, pages 61–77. Springer, 2012.

[KR12] Gary Kenworthy and Pankaj Rohatgi. Mobile Device Security: The case for side channel resistance. Presentation at MOST'12, 2012. `http://mostconf.org/2012/slides/21.pdf`.

[Kru04] Ralf Krueger. Application Note XAPP766: Using High Security Features in Virtex-II Series FPGAs. Technical report, Xilinx, 2004. `http://www.xilinx.com/support/documentation/application_notes/xapp766.pdf`.

[KSP10] Timo Kasper, Michael Silbermann, and Christof Paar. All You Can Eat or Breaking a Real-World Contactless Payment System. In *Financial Cryptography and Data Security 2010*, LNCS. Springer, 2010.

[KvMOP11] Timo Kasper, Ingo von Maurich, David Oswald, and Christof Paar. Chameleon: A versatile emulator for contactless smartcards. In Kyung-Hyune Rhee and DaeHun Nyang, editors, *Information Security and Cryptology – ICISC'10*, volume 6829 of *LNCS*, pages 189–206. Springer, 2011.

[KW05] Ziv Kfir and Avishai Wool. Picking virtual pockets using relay attacks on contactless smartcard systems. Cryptology ePrint Archive, Report 2005/052, 2005. http://eprint.iacr.org.

[KW06] Ilan Kirschenbaum and Avishai Wool. How to Build a Low-Cost, Extended-Range RFID Skimmer. In *USENIX Security Symposium*. USENIX Association, 2006.

[Lan] Langer EMV-Technik. Preamplifier PA 303. Website as of April 2013. http://www.langer-emv.de/en/products/disturbance-emission/preamplifier/pa-203-303/devices-data/.

[LKLRP07] Y. Liu, T. Kasper, K. Lemke-Rust, and C. Paar. E-Passport: Cracking Basic Access Control Keys. In *Proc. of OTM'07, Part II*, volume 4804 of *LNCS*, pages 1531–1547. Springer, 2007.

[LLG09] Ruilin Li, Chao Li, and Chunye Gong. Differential Fault Analysis on SHACAL-1. In *Proceedings of the 2009 Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC'09*, pages 120–126, Washington, DC, USA, 2009. IEEE Computer Society.

[Mag13a] Magistrat hl. m. Prahy. Opencard – Frequently Asked Questions. Website, April 2013. http://opencard.praha.eu/jnp/en/faq/index.html.

[Mag13b] Magistrat hl. m. Prahy. Opencard – Your convenience card. Website, April 2013. http://opencard.praha.eu/jnp/en/home/index.html.

[Mah36] P. C. Mahalanobis. On the Generalised Distance in Statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, April 1936.

[Man02] Stefan Mangard. A Simple Power-Analysis (SPA) attack on implementations of the AES key expansion. In *ICISC'02*, volume 2587 of *LNCS*, pages 343–358. Springer-Verlag, 2002.

[Max13] Maxim Integrated. DS28E01-100 1Kb Protected 1-Wire EEPROM with SHA-1 Engine. Website, April 2013. http://www.maximintegrated.com/datasheet/index.mvp/id/4766.

[MBCB+11] Uwe Meyer-Bäase, Encarni Castillo, Guillermo Botella, L. Parrilla, and Antonio Garcia. Intellectual property protection (IPP) using obfuscation in C, VHDL, and Verilog coding. *SPIE Proceedings Independent Component Analyses, Wavelets, Neural Networks, Biosystems, and Nanoengineering IX*, 8058, 2011.

[MBKP11] Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. On the vulnerability of FPGA bitstream encryption against power analysis attacks: Extracting keys from Xilinx Virtex-II FPGAs. In *ACM CCS'11*, pages 111–124. ACM, 2011.

[MDS99] Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Investigations of Power Analysis Attacks on Smartcards. In *USENIX Workshop on Smartcard Technology*, pages 151–162, 1999.

[Mes00] Thomas S. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In Çetin K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'00*, volume 1965 of *LNCS*, pages 238–251. Springer, 2000.

[Met13] Metropolitan Transportation Commission. Clippercard. Website, April 2013. `https://www.clippercard.com/ClipperWeb/index.do`.

[Mic09] Microchip Technology Inc. PIC16F882/883/884/886/887 Data Sheet, 2009. available at `http://ww1.microchip.com/downloads/en/devicedoc/41291f.pdf`.

[MKP12] Amir Moradi, Markus Kasper, and Christof Paar. Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures - An Analysis of the Xilinx Virtex-4 and Virtex-5 Bitstream Encryption Mechanism. In *CT-RSA'12*, volume 7178 of *LNCS*, pages 1–18. Springer, 2012.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.

[MOPS13] Amir Moradi, David Oswald, Christof Paar, and Pawel Swierczynski. Side-channel attacks on the bitstream encryption mechanism of Altera Stratix II: facilitating black-box analysis using software reverse-engineering. In *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays – FPGA'13*, pages 91–100, New York, NY, USA, 2013. ACM.

[MRL+06] Yannick Monnet, Marc Renaudin, Régis Leveugle, Christophe Clavier, and Pascal Moitrel. Case Study of a Fault Attack on Asynchronous DES Crypto-Processors. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *FDTC'06*, volume 4236 of *LNCS*, pages 88–97. Springer, 2006.

[MTMM07] Robert McEvoy, Michael Tunstall, Colin C. Murphy, and William P. Marnane. Differential power analysis of hmac based on sha-2, and countermeasures. In *Proceedings of the 8th international conference on Information security applications*, WISA'07, pages 317–332. Springer, 2007.

[MvOV96] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[Nat09] National Institute of Advanced Industrial Science and Technology (AIST). *Side-Channel Attack Standard Evaluation Board SASEBO-GII Specification*, 1.01 edition, 2009.

[NESP08] Karsten Nohl, David Evans, Starbug, and Henryk Plötz. Reverse-Engineering a Cryptographic RFID Tag. In *USENIX Security Symposium*, pages 185–194. USENIX Association, 2008.

[NISa] NIST. FIPS 180-4 Secure Hash Standard (SHS). `http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf`.

[NISb] NIST. FIPS 197 Advanced Encryption Standard (AES). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[NISc] NIST. FIPS 46-3 Data Encryption Standard (DES). http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

[NIS01] NIST. *Recommendation for Block 2001 Edition Cipher Modes of Operation*, 2001. http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.

[NR08] Jean-Baptiste Note and Éric Rannaud. From the bitstream to the netlist. In *Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays – FPGA'08*, pages 264–264. ACM, 2008.

[NT11] Karsten Nohl and Christopher Tarnovsky. Reviving smart card analysis. Presentation at Chaos Communication Camp, 2011. http://events.ccc.de/camp/2011/Fahrplan/attachments/1888_SRLabs-Reviving_Smart_Card_Analysis.pdf.

[NXP04] NXP. *Mifare DESFire Contactless Multi-Application IC with DES and 3DES Security MF3ICD40*, April 2004.

[NXP06] NXP. mifare DESFire Security. Presentation slides, 2006.

[NXP07] NXP. AN200701: ISO/IEC 14443 Eavesdropping and Activation Distance. Technical report, 2007.

[NXP09] NXP. *Mifare Ultralight C Product Short Datasheet*, May 2009. http://www.nxp.com/documents/short_data_sheet/MF0ICU2_SDS.pdf.

[NXP10a] NXP. AN10922 – Symmetric key diversifications. Application note, 2010. http://www.nxp.com/documents/application_note/AN10922.pdf.

[NXP10b] NXP. *Mifare DESFire EV1 Contactless Multi-Application IC Datasheet*, December 2010. http://www.nxp.com/documents/short_data_sheet/MF3ICDX21_41_81_SDS.pdf.

[Och06] Karlheinz Ochs. Transmission of Digital Signals. Lecture notes, 2006.

[OP11] David Oswald and Christof Paar. Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In *Cryptographic Hardware and Embedded Systems – CHES'11*, volume 6917 of *LNCS*, pages 207–222. Springer, 2011.

[OP13] David Oswald and Christof Paar. Improving Side-Channel Analysis with Optimal Linear Transforms. In Stefan Mangard, editor, *Smart Card Research and Advanced Applications – CARDIS'12*, volume 7771 of *LNCS*, pages 219–233. Springer, 2013.

[ORP13] David Oswald, Bastian Richter, and Christof Paar. Side-Channel Attacks on the Yubikey 2 One-Time Password Generator. to appear at RAID'13, 2013.

[OS07] Yossef Oren and Adi Shamir. Remote Password Extraction from RFID Tags. *IEEE Transactions on Computers*, 56(9):1292–1296, 2007. http://iss.oy.ne.ro/RemotePowerAnalysisOfRFIDTags.

[Osm13] Osmocom Project. rtl-sdr. Website, May 2013. http://sdr.osmocom.org/trac/wiki/rtl-sdr.

[OSS+13] David Oswald, Daehyun Strobel, Falk Schellenberg, Timo Kasper, and Christof Paar. When Reverse-Engineering Meets Side-Channel Analysis – Digital Lockpicking in Practice. In *Selected Areas in Cryptography – SAC'13*, LNCS. Springer, 2013.

[Osw09] David Oswald. Development of an Integrated Environment for Side Channel Analysis and Fault Injection. Diploma thesis, Ruhr-University Bochum, September 2009.

[Osw12a] David Oswald. Practical Side-Channel Attacks against Contactless Smart Cards. Presentation at ISCTURKEY'12, May 2012.

[Osw12b] David Oswald. Pushing the Limits: Side-Channel Attacks on Mifare DESFire MF3ICD40 for 300 USD. Presentation at the ACM S3 Workshop 2012, August 2012.

[PA13] Kenny Paterson and Nadhem AlFardan. On the Security of RC4 in TLS. Website, March 2013. http://www.isg.rhul.ac.uk/tls/.

[PHF08] Thomas Plos, Michael Hutter, and Martin Feldhofer. Evaluation of Side-Channel Preprocessing Techniques on Cryptographic-Enabled HF and UHF RFID-Tag Prototypes. In Sandra Dominikus, editor, *Workshop on RFID Security – RFIDSec'08*, pages 114–127, 2008.

[Pic08] Pico Technology. PicoScope 5200 USB PC Oscilloscopes, 2008. http://www.picotech.com/picoscope5200-specifications.html.

[Plo09] Thomas Plos. Evaluation of the Detached Power Supply as Side-Channel Analysis Countermeasure for Passive UHF RFID Tags. In Marc Fischlin, editor, *Topics in Cryptology – CT-RSA'09*, volume 5473 of *LNCS*, pages 444–458. Springer, 2009.

[PM05] Thomas Popp and Stefan Mangard. Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES'05*, volume 3659 of *LNCS*, pages 172–186. Springer, 2005.

[PN09] Henrik Plötz and Karsten Nohl. Legic Prime: Obscurity in Depth. Presentation at the 26th Chaos Communication Congress, 2009. http://events.ccc.de/congress/2009/Fahrplan/attachments/1506_legic-slides.pdf.

[PO95] Bart Preneel and Paul C. Oorschot. MDx-MAC and Building Fast MACs from Hash Functions. In Don Coppersmith, editor, *Advances in Cryptology – CRYPT0'95*, volume 963 of *LNCS*, pages 1–14. Springer, 1995.

[PP10] Christof Paar and Jan Pelzl. *Understanding Cryptography – A Textbook for Students and Practitioners*. Springer, 2010.

[RCT06] Melanie R. Rieback, Bruno Crispo, and Andrew S. Tanenbaum. Is Your Cat Infected with a Computer Virus? In *Conference on Pervasive Computing and Communications (PerCom)*, pages 169–179. IEEE Computer Society, 2006.

[Rea13] Realtek Semiconductors Corp. RTL2832U General Description. Website, April 2013. `http://www.realtek.com.tw/products/productsView.aspx?Langid=1&PNid=22&PFid=35&Level=4&Conn=3&ProdID=257`.

[RNP10] Andreas Rohr, Karsten Nohl, and Henryk Plötz. Establishing Security Best Practices in Access Control. September 2010. `www.srlabs.de/pub/acs`.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably Secure Higher-Order Masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES'10*, volume 6225 of *LNCS*, pages 413–427. Springer, 2010.

[RS09] Mathieu Renauld and François-Xavier Standaert. Algebraic Side-Channel Attacks. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *5th International Conference on Information Security and Cryptology – Inscrypt'09*, volume 6151 of *LNCS*, pages 393–410. Springer, 2009.

[RSVC+11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In *30th Annual international conference on Theory and applications of cryptographic techniques – EUROCRYPT'11*, LNCS, pages 109–128. Springer, 2011.

[SA08] François-Xavier Standaert and Cedric Archambeau. Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES'08*, volume 5154 of *LNCS*, pages 411–425. Springer, 2008.

[SBS66] Mischa Schwartz, William R. Bennett, and Seymour Stein. *Communication Systems and Techniques*. Wiley, 1966.

[Sch10] Falk Schellenberg. Comparing Power and Electromagnetic Analysis of Embedded Devices. Bachelor's thesis, Ruhr-University Bochum, Juli 2010.

[SDK+13] Daehyun Strobel, Benedikt Driessen, Timo Kasper, Gregor Leander, David Oswald, Falk Schellenberg, and Christof Paar. Fuming Acid and Cryptanalysis: Handy Tools for Overcoming a Digital Locking and Access Control System. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO'13*, volume 8042 of *LNCS*, pages 147–164. Springer, 2013.

[Sha06] K. Sam Shanmugam. *Digital & Analog Communication Systems*, chapter 8.3.2. Wiley-India, 2006.

[Sima] SimonsVoss Technologies AG. SimonsVoss posts record sales yet again in 2011. available at http://www.simons-voss.us/Record-sales-in-2011.1112.0.html?&L=6, as of October 7, 2013.

[Simb] SimonsVoss Technologies AG. Wave Net Funknetzwerk 3065. available at http://www.simons-voss.de/Wave-Net-Funknetzwerk.74.0.html.

[Sim13] SimonsVoss Technologies AG. Infocenter – Downloads, 2013. available at http://www.simons-voss.com/Downloads.45.0.html?&L=1.

[SLFP04] Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar. A Collision-Attack on AES Combining Side channel- and Differential-Attack. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES'04*, volume 3156 of *LNCS*, pages 163–175. Springer, 2004.

[Smi97] Ph.D. Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1st edition, 1997.

[Smi07] Julius O. Smith. *Introduction To Digital Filters With Audio Applications*, chapter General LTI Filter Matrix. Center for Computer Research in Music and Acoustics, 2007. http://www.dsprelated.com/dspbooks/filters/.

[SMY09] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT'09*, volume 5479 of *LNCS*, pages 443–461. Springer, 2009.

[SNK+12] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. Simple Photonic Emission Analysis of AES. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES'12*, volume 7428 of *LNCS*, pages 41–57. Springer, 2012.

[Sok01] Nathan O. Sokal. Class-E RF Power Amplifiers. *QEX Commun. Quart*, pages 9–20, 2001. http://www.electro-tech-online.com/custompdfs/2011/09/010102qex009.pdf.

[Sou13a] Sourceforge. GIAnT (Generic Implementation ANalysis Toolkit). Website, April 2013. https://sf.net/projects/giant/.

[Sou13b] Sourceforge. reader_14443: Customized RFID Reader for Contactless Smartcards. Website, April 2013. http://sourceforge.net/projects/reader14443/.

[ST04] Adi Shamir and Eran Tromer. Acoustic cryptanalysis – On nosy people and noisy machines. Website, 2004. http://tau.ac.il/~tromer/acoustic/.

[STA+10] Jörn-Marc Schmidt, Michael Tunstall, Roberto Maria Avanzi, Ilya Kizhvatov, Timo Kasper, and David Oswald. Combined Implementation Attack Resistant Exponentiation. In *LATINCRYPT'10*, volume 6212 of *LNCS*, pages 305–322. Springer, 2010.

[Sta11] State Government Victoria. myki. Website, March 2011. http://www.myki.com.au/.

[Str07] Stratix II Device Handbook, Volume 1. Technical report, Altera, 2007. http://www.altera.com/literature/hb/stx2/stratix2_handbook.pdf.

[Sun] Sunplus Innovation Technology Inc. http://www.sunplusit.com.

[SW12a] Sergei Skorobogatov and Christopher Woods. Breakthrough silicon scanning discovers backdoor in military chip. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES'12*, volume 7428 of *LNCS*, pages 23–40. Springer, 2012.

[SW12b] Sergei Skorobogatov and Christopher Woods. In the blink of an eye: There goes your AES key. Cryptology ePrint Archive, Report 2012/296, 2012. http://eprint.iacr.org/.

[Tar10] Christopher Tarnovsky. Deconstructing a 'Secure' Processor. Presentation at Black Hat USA, 2010. http://www.blackhat.com/html/bh-dc-10/bh-dc-10-speaker_bios.html#Tarnovsky.

[TAV02] K. Tiri, M. Akmal, and I. Verbauwhede. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Proceedings of the 28th European Solid-State Circuits Conference – ESSCIRC'02.*, pages 403–406, 2002.

[Tel13] Teledyne LeCroy. WavePro 715Zi-A. Website, April 2013. http://teledynelecroy.com/oscilloscope/oscilloscopemodel.aspx?modelid=4716.

[The] The MathWorks, Inc. MATLAB R2011b Documentation, Optimization Toolbox, fminunc. Website as of February 2012. http://www.mathworks.de/help/toolbox/optim/ug/fminunc.html.

[The10] The Prague Post. City faces big fine for OpenCard. December 2010. http://www.praguepost.com/news/6622-city-faces-big-fine-for-opencard.html.

[Tse05] Chen Wei Tseng. Lock Your Designs with the Virtex-4 Security Solution. XCell Journal, Xilinx, 2005.

[TV03] Kris Tiri and Ingrid Verbauwhede. Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology. In Colin D. Walter, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'03*, volume 2779 of *LNCS*, pages 125–136. Springer, 2003.

[Uni13] United Nations Office on Drugs and Crime. Counterfeit Goods - A bargain or a costly mistake? Fact Sheet, 2013. http://www.unodc.org/documents/toc/factsheets/TOC12_fs_counterfeit_EN_HIRES.pdf.

[Vam12] Loredana Vamanu. Formal Analysis of Yubikey. Master's thesis, INRIA, 2012.

[VCGRS13] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography – SAC'12*, volume 7707 of *LNCS*, pages 390–406. Springer, 2013.

[VCMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: a comprehensive study with cautionary note. In *Proceedings of the 18th international conference on The Theory and Application of Cryptology and Information Security – ASIACRYPT'12*, LNCS, pages 740–757. Springer, 2012.

[VGB12] Roel Verdult, Flavio D. Garcia, and Josep Balasch. Gone in 360 seconds: Hijacking with Hitag2. In *USENIX Security Symposium*, pages 237–252. USENIX Association, August 2012. `https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final95.pdf`.

[Visa] Vishay Semiconductors, Inc. *BAT43 Schottky Diode Datasheet*.

[Visb] Vishay Semiconductors, Inc. *BAT48 Schottky Diode Datasheet*.

[vWWB11] Jasper G. J. van Woudenberg, Marc F. Witteman, and Bram Bakker. Improving Differential Power Analysis by Elastic Alignment. In *Topics in Cryptology – CT-RSA'11*, volume 6558 of *LNCS*, pages 104–119. Springer, 2011.

[Wal04] Colin D. Walter. Simple power analysis of unified code for ecc double and add. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES'04*, volume 3156 of *LNCS*, pages 191–204. Springer, 2004.

[Wei10] Eric W. Weisstein. Variance. Mathworld - A Wolfram Web Resource, December 2010. `http://mathworld.wolfram.com/Variance.html`.

[WGP04] Thomas J. Wollinger, Jorge Guajardo, and Christof Paar. Security on FPGAs: State-of-the-art implementations and attacks. *ACM Transactions in Embedded Computing Systems (TECS)*, 3(3):534–574, 2004.

[WOS12] Carolyn Whitnall, Elisabeth Oswald, and François-Xavier Standaert. The myth of generic DPA... and the magic of learning. Cryptology ePrint Archive, Report 2012/256, 2012. `http://eprint.iacr.org/`.

[Xil13] Xilinx, Inc. *7 Series FPGAs Configuration User Guide*, v 1.6 edition, January 2013. `http://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf`.

[Yub] Yubico. Yubico Reference Customers: Department of Defense. http://www.yubico.com/about/reference-customers/department-defence/.

[Yub12] Yubico. *The YubiKey Manual*. Yubico, May 2012.

[Yub13a] Yubico. Website, April 2013. `www.yubico.com`.

[Yub13b] Yubico. Download of personalisation tool. Website, April 2013. http://www.yubico.com/products/services-software/personalization-tools/.

[Yub13c] Yubico. How YubiKeys are manufactured. Website, April 2013. https://www.youtube.com/watch?v=s8_I1-ErZSQ.

[Yub13d] Yubico. YubiKey NEO. Website, February 2013. `http://www.yubico.com/products/yubikey-hardware/yubikey-neo/`.

[Yub13e] Yubico. *Yubikey Security Evaluation Version 2.0*, April 2013.

[Zon11] Andrew Zonenberg. Microchip PIC12F683 teardown, 2011. available at `siliconexposed.blogspot.de/2011/03/microchip-pic12f683-teardown.html`.

[ZTE13] ZTEX GmbH. USB FPGA Module 1.11: Spartan 6 FPGA Board with USB 2.0 Microcontroller and 64 MByte DDR SDRAM. Website, February 2013. `http://www.ztex.de/usb-fpga-1/usb-fpga-1.11.e.html`.

List of Abbreviations

3DES Triple DES

ADC Analog to Digital Converter

AES Advanced Encryption Standard

APDU Application Protocol Data Unit

API Application Programming Interface

ASCA Algebraic Side-Channel Analysis

ASK Amplitude-Shift Keying

ASIC Application Specific Integrated Circuit

ATR Answer To Reset

BAC Basic Access Control

BPSK Binary Phase Shift Keying

CBC Cipher Block Chaining

CC Common Criteria

CCA Canonical Correlation Analysis

CIA Combined Implementation Attacks

CMOS Complementary Metal Oxide Semiconductor

CPA Correlation Power Analysis

CPU Central Processing Unit

CRC Cyclic Redundancy Check

CRT Chinese Remainder Theorem

CTR Counter (mode of operation)

DAC Digital-Analog Converter

DES Data Encryption Standard

DEMA Differential Electro-Magnetic Analysis

DFA Differential Frequency Analysis

DFT Discrete Fourier Transform

DoM Difference of Means

DoS Denial-of-Service

DPA Differential Power Analysis

DRAM Dynamic Random Access Memory

DSO Digital Storage Oscilloscope

DSP Digital Signal Processing

DST Digital Signature Transponder

DTW Dynamic Time Warping

DUT Device Under Test

DVB-T Digital Video Broadcasting – Terrestrial

ECC Elliptic Curve Cryptography

EDE Encrypt-Decrypt-Encrypt (mode of operation)

EEPROM Electrically Erasable Programmable Read-Only Memory

EM Electro-Magnetic

FFT Fast Fourier Transform

FI Fault Injection

FIB Focused Ion Beam

FIFO First In First Out (memory)

FIR Finite Impulse Response

FPGA Field Programmable Gate Array

FSK Frequency Shift Keying

GIAnT Generic Implementation Analysis Toolkit

GPIO General Purpose I/O

GPL GNU General Public License

GPS Global Positioning System

HD Hamming Distance

HDL Hardware Description Language

HF High Frequency

HID Human Interface Device

HMAC Hash-based Message Authentication Code

HOTP HMAC-based One Time Password

HW Hamming Weight

IC Integrated Circuit

ISM Industrial, Scientific, and Medical (frequencies)

IP Intellectual Property

IV Initialization Vector

JTAG Joint Test Action Group

LF Low Frequency

LSB Least Significant Bit

LSByte Least Significant Byte

LUT Look-Up Table

MAC Message Authentication Code

MIA Mutual Information Analysis

MITM Man-In-The-Middle

MOSFET Metal-Oxide Semiconductor Field-Effect Transistor

MSB Most Significant Bit

MSByte Most Significant Byte

µC Microcontroller

NFC Near Field Communication

NRZ Non-Return-to-Zero (encoding)

NVM Non-Volatile Memory

OATH Initiative of Open Authentication

OOK On-Off-Keying

OP Operational Amplifier

OTP One-Time Password

PC Personal Computer

PCA Principal Component Analysis

PCB Printed Circuit Board

PEA Photonic Emission Analysis

PKI Public Key Infrastructure

PS Passive Serial (mode)

RAM Random Access Memory

RF Radio Frequency

RFID Radio Frequency IDentification

RKE Remote Keyless Entry

RNG Random Number Generator

RSA Rivest Shamir and Adleman

SAM Square-and-Multiply

SCA Side-Channel Analysis

SDR Software-Defined Radio

SEM Scanning Electron Microscopy

SNR Signal to Noise Ratio

SHA-1 Secure Hash Algorithm 1

SMA SubMiniature version A (connector)

SOF Start Of Frame

SPA Simple Power Analysis

SPOF Single Point of Failure

SRAM Static Random Access Memory

TA Template Attack

TEM Transmission Electron Microscopy

TNR Trace-to-Noise Ratio

TWI Two Wire Interface

SPI Serial Peripheral Interace

UHF Ultra High Frequency

UID Unique Identifier

USB Universal Serial Bus

USRP2 Universal Software Radio Peripheral (version 2)

UV-C Ultraviolet-C (light)

VHDL VHSIC (Very High Speed Integrated Circuit) Hardware Description Language

List of Figures

List of Tables

About the Author

Personal Data

Name: David Frederic Oswald

Address: Chair for Embedded Security, ID 2/605, Universitätsstr. 150, 44801 Bochum, Germany

E-Mail: david.oswald@rub.de

Date of birth: December 24, 1984

Place of birth: Essen, Germany

Education

Ruhr-University Bochum

October 2009–Present: Ph.D. candidate, Chair for Embedded Security

September 2009: "Dipl.-Ing. IT Security" [equivalent to an M.Sc.], final grade: 98%

March 2009–September 2009: Diploma thesis "Development of an Integrated Environment for Side Channel Analysis and Fault Injection", Chair for Embedded Security

October 2004–September 2009: Combined B.Sc./M.Sc. program "IT Security", Department of Electrical Engineering and Information Technology

Viktoria High School, Essen, Germany

June 2004: General qualification for university entrance (Abitur)

Professional Experience

Work Experience

October 2012: Founder of Kasper & Oswald GmbH (`http://www.kasper-oswald.de`), specialized in security-related aspects of embedded systems

July 2011–September 2011: Internship Cryptography Research Inc., San Francisco, USA, R&D department: Side-Channel Analysis of an AES Implementation

September 2008–January 2009: Internship Rohde & Schwarz GmbH & Co. KG, Munich, Germany, R&D department: Efficient Implementation of a Mixed-Radix FFT in VHDL

Research and Industry Projects

July 2011–October 2011: Project manager for the side-channel evaluation of commercial products for German industry

April 2009–October 2010: Responsible project manager of the study "RFID Security" conducted for a German federal organization, Chair for Embedded Security, Prof. Dr.-Ing. C. Paar

Academic Awards

January 2010: Department award: "Best in graduation class"

January 2007: Department award: "Academic excellence in first and second year"

March 2007: "Best students in software engineering", Prof. Dr.-Ing. habil. H. Balzert

Publications

The author of this thesis has worked in several interleaved research areas. In the following, all the formal and informal publications are listed where the author of this thesis was either the lead author (■) or one of the co-authors (⧫) .

Peer-Reviewed Publications in Journals

■ Timo Kasper, David Oswald, and Christof Paar. A Versatile Framework for Implementation Attacks on Cryptographic RFIDs and Embedded Devices. In Marina Gavrilova, C. Tan, and Edward Moreno, editors, *Transactions on Computational Science X*, volume 6340 of *LNCS*, pages 100–130. Springer, 2010.

Peer-Reviewed Publications in Proceedings of International Conferences

■ David Oswald, Bastian Richter, and Christof Paar. Side-Channel Attacks on the Yubikey 2 One-Time Password Generator. To appear in the LNCS proceedings of RAID'13, 2013.

■ David Oswald, Daehyun Strobel, Falk Schellenberg, Timo Kasper, and Christof Paar. When Reverse-Engineering Meets Side-Channel Analysis – Digital Lockpicking in Practice. In *Selected Areas in Cryptography – SAC'13*, LNCS. Springer, 2013.

⧫ Daehyun Strobel, Benedikt Driessen, Timo Kasper, Gregor Leander, David Oswald, Falk Schellenberg, and Christof Paar. Fuming Acid and Cryptanalysis: Handy Tools for Overcoming a Digital Locking and Access Control System. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO'13*, volume 8042 of *LNCS*, pages 147–164. Springer, 2013.

⧫ Timo Kasper, Alexander Kühn, David Oswald, Christian Zenger, and Christof Paar. Rights Management with NFC Smartphones and Electronic ID Cards: A Proof of Concept for Modern Car Sharing. In *Workshop on RFID Security – RFIDSec'13*, LNCS, Graz, Austria, July 2013. Springer.

■ David Oswald and Christof Paar. Improving Side-Channel Analysis with Optimal Linear Transforms. In Stefan Mangard, editor, *Smart Card Research and Advanced Applications – CARDIS'12*, volume 7771 of *LNCS*, pages 219–233. Springer, 2013.

- Amir Moradi, David Oswald, Christof Paar, and Pawel Swierczynski. Side-channel attacks on the bitstream encryption mechanism of Altera Stratix II: facilitating black-box analysis using software reverse-engineering. In *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays – FPGA'13*, pages 91–100, New York, NY, USA, 2013. ACM.

- Timo Kasper, David Oswald, and Christof Paar. Side-channel analysis of cryptographic RFIDs with analog demodulation. In Ari Juels and Christof Paar, editors, *7th International Workshop on RFID Security and Privacy – RFIDSec'11*, volume 7055 of *LNCS*, pages 61–77. Springer, 2012.

- David Oswald and Christof Paar. Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In *CHES 2011*, volume 6917 of *LNCS*, pages 207–222. Springer, 2011.

- Timo Kasper, Ingo von Maurich, David Oswald, and Christof Paar. Chameleon: A versatile emulator for contactless smartcards. In Kyung-Hyune Rhee and DaeHun Nyang, editors, *Information Security and Cryptology – ICISC 2010*, volume 6829 of *LNCS*, pages 189–206. Springer, 2011.

- Jörn-Marc Schmidt, Michael Tunstall, Roberto Maria Avanzi, Ilya Kizhvatov, Timo Kasper, and David Oswald. Combined Implementation Attack Resistant Exponentiation. In *LATINCRYPT*, volume 6212 of *LNCS*, pages 305–322. Springer, 2010.

- Timo Kasper, David Oswald, and Christof Paar. Wireless security threats: Eavesdropping and detecting of active RFIDs and remote controls in the wild. In *19th International Conference on Software, Telecommunications and Computer Networks – SoftCOM 2011*, pages 1–6, 2011.

- Timo Kasper, David Oswald, and Christof Paar. EM Side–Channel Attacks on Commercial Contactless Smartcards Using Low-Cost Equipment. In Heung Youl Youm and Moti Yung, editors, *10th International Workshop on Information Security Applications – WISA 2009*, LNCS, pages 79–93. Springer, 2009.

Publications and Presentations at National Conferences & Workshops

- David Oswald. Pushing the Limits: Side-Channel Attacks on Mifare DESFire MF3ICD40 for 300 USD. Presentation at the ACM S3 Workshop 2012, Istanbul, Turkey, August 2012.

- David Oswald. Practical Side-Channel Attacks against Contactless Smart Cards. Presentation at ISCTURKEY 2012, Ankara, Turkey, May 2012.

- Timo Kasper, David Oswald, and Christof Paar. Security of Wireless Embedded Devices in the Real World. In Norbert Pohlmann, Helmut Reimer, and Wolfgang Schneider, editors, *Information Security Solutions (ISSE) 2011 – Securing Electronic Business Processes*, pages 174–189. Vieweg+Teubner, 2011.

- Timo Kasper, Ingo von Maurich, David Oswald, and Christof Paar. Cloning Cryptographic RFID Cards for 25$. WISSec, Nijmegen, the Netherlands, 2010

Magazines

- Timo Kasper, David Oswald, and Christof Paar. Seitenkanalanalyse kontaktloser Smart-Cards. In Datenschutz und Datensicherheit - DuD, Ausgabe 11/2011.

Participation in Selected Conferences & Workshops

- FPGA 2013 (Monterey, USA)
- CARDIS 2012 (Graz, Austria)
- CHES 2012 (Leuven, Belgium)
- ISCTURKEY 2012 (Ankara, Turkey)
- Escar 2011 (Dresden, Germany)
- CHES 2011 (Nara, Japan)
- FDTC 2011 (Nara, Japan)
- RFIDSec 2011 (Amherst, USA)
- Ecrypt 2 Summer School 2011 (Albena, Bulgaria)
- ICISC 2010 (Seoul, Korea)
- CRYPTO 2010 (Santa Barbara, USA)
- CHES 2010 (Santa Barbara, USA)
- FDTC 2010 (Santa Barbara, USA)
- RFIDSec 2009 (Leuven, Belgium)
- FDTC 2009 (Leuven, Belgium)
- CHES 2009 (Lausanne, Switzerland)