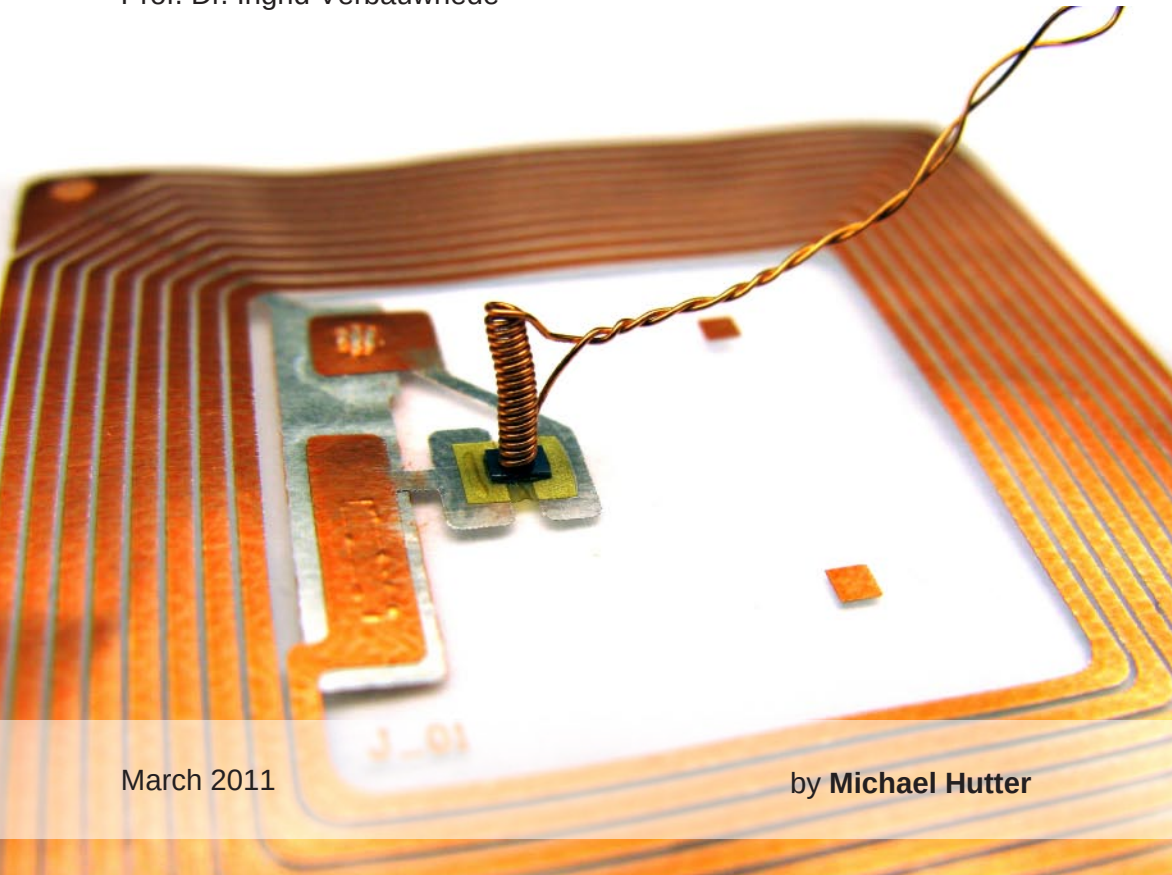
Graz University of Technology
Faculty of Computer Science
Institute for Applied Information Processing and Communications
IAIK

Secure and Efficient Implementation of RFID Tags Supporting Cryptography

A PhD Thesis Presented to the Faculty of Computer Science in Fulfillment of the Requirements for the PhD Degree

Assessors:
Prof. Dr. Karl Christian Posch
Prof. Dr. Ingrid Verbauwhede



March 2011

by **Michael Hutter**

Secure and Efficient Implementation of RFID Tags Supporting Cryptography

by

Michael Hutter

A PhD Thesis
Presented to the Faculty of Computer Science in Partial Fulfillment of the Requirements for the PhD Degree

Assessors

Prof. Dr. Karl Christian Posch (TU Graz, Austria)
Prof. Dr. Ingrid Verbauwhede (KU Leuven, Belgium)

March 2011

Institute for Applied Information Processing and Communications (IAIK)
Faculty of Computer Science
Graz University of Technology, Austria

Abstract

Radio Frequency Identification (RFID) is a wireless communication technology that has gained a lot of importance in the last decade. This thesis focuses on security aspects of the RFID technology and has been structured into two main parts. The first part analyzes the susceptibility of RFID devices against different implementation attacks. Such attacks make use of physical properties and behavior of electronic devices to extract secret information. We evaluate their vulnerability by presenting results of successful side-channel attacks as well as fault-analysis attacks. Furthermore, we investigate public-key enabled RFID devices that provide cryptographic services like proof-of-origin authentication by digitally signing of data. We present practical attacks on such devices that reveal the private key by power and electromagnetic analysis.

The second part makes use of the findings in part one in order to design a low-resource cryptographic processor for passive RFID tags. The design meets fierce requirements concerning chip area (costs), power consumption (contactless operation), and security. First, we introduce new formulæ for Elliptic Curve Cryptography (ECC) that allow an efficient implementation on resource-constraint devices using Co-Z coordinate representation over prime fields. Thereafter, we present the design of a hardware processor that applies the new formulæ together with several optimization techniques to implement an efficient and secure RFID processor resistant to most of the presented attacks. We combine both symmetric as well as asymmetric primitives into one piece of silicon by implementing AES-128 and ECDSA using the recommended NIST elliptic curve over $\mathbb{F}_{p192}$.

The outcomes of the thesis are as follows. First, we show that passive RFID devices are vulnerable to side-channel attacks as well as fault attacks. Most of our experiments can be performed with low cost and simple measurement setups. By evaluating the vulnerability of such devices, we emphasize the need of appropriate countermeasures for passive RFID devices that prevent impersonation through cloning of tags, forging of digital signatures, or the ability of adversaries to provide a proof of origin of RFID-labeled counterfeited goods, for instance. Second, we present a cryptographic processor for passive RFID devices that implements strong cryptography. We apply new formulæ for ECC over $\mathbb{F}_p$ that improve the state of the art in low-resource implementations in terms of both memory and speed.

Acknowledgements

I would like to thank all people who supported me in writing this thesis. Special thanks go to my colleagues Jörn-Marc Schmidt and Thomas Plos who have been good friends during my PhD studies and with which I share many memorable moments during our work on research projects and non-technical events.

I would like to thank Reinhard Posch for the possibility to do my PhD study at IAIK. As a member of the VLSI group, I especially thank Manfred Aigner who merits special acknowledgements for his support in any work and project-related issues. I also would like to thank Karl-Christian Posch who has been the advisor of this thesis and who has been a great mentor in my work as a teaching assistant at TU Graz. I also would like to thank Ingrid Verbauwhede as external assessor of the thesis and for making the trip to Graz. Furthermore, I thank Stefan Mangard, Marcel Medwed, Christoph Herbst, Marc Joye, and Yannick Sierra for many helpful discussions in different projects.

Many thanks go to my colleagues at IAIK. Special thanks go to Johannes Wolkerstorfer who supported me in any hardware and ECC-related topics and Martin Feldhofer for his valuable inputs not only during our collaboration within the *CRYPTA* project. I also would like to thank Alexander Szekely for many fruitful discussions and collaboration time during the *QCC* project. Furthermore, I thank Mario Kirschbaum, Sandra Dominikus, Erich Wenger, Stefan Tillich, Thomas Popp, Daniel Hein, Markus Pelnar, and Christoph Nagl and all others who reviewed parts of this work.

Finally, I would like to thank my family and friends who supported me during my studies. Without them, the work would not have been possible. Special thanks go to my parents, my brother David and my sister Margret. I also thank my friends and study colleagues Thomas Zefferer, Andreas Weitzer, Stefan Kremser, and Matthias Schröfelbauer.

Special thanks go to my girlfriend Sandra for her understanding and encouragement during my studies and work on this thesis.

Michael Hutter
Graz, March 2011

Table of Contents

List of Tables

List of Figures

List of Publications

In Refereed Journals

1. Michael Hutter, Thomas Plos, Martin Feldhofer, On the Security of RFID Devices against Implementation Attacks. In *International Journal of Security and Networks*, 5(2/3):106-118, March 2010.

2. Ronald Tögl, Michael Hutter, An Approach to Introducing Locality in Remote Attestation using Near Field Communications. In *Journal of Supercomputing* 54:1-21, March 2010.

In Refereed Conference Proceedings

1. 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 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 320-333. Springer, September 2007.

2. Jörn-Marc Schmidt and Michael Hutter. Optical and EM Fault-Attacks on CRT-based RSA: Concrete Results. In *Austrochip 2007, 15th Austrian Workhop on Microelectronics, 11 October 2007, Graz, Austria, Proceedings*, pages 61-67, October 2007.

3. Thomas Plos, Michael Hutter, and Christoph Herbst. Enhancing Side-Channel Analysis with Low-Cost Shielding Techniques. *In Proceedings of Austrochip 2008, Linz, Austria, October 8, 2008, Proceedings*, pages 90-95. October 2008.

4. Michael Hutter, Jörn-Marc Schmidt, and Thomas Plos. RFID and Its Vulnerability to Faults. *In Cryptographic Hardware and Embedded Systems CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008, Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 363-379. Springer, August 2008.

5. Thomas Lorünser, Edwin Querasser, Thomas Matyus, Momtchil Peev, Johannes Wolkerstorfer, Michael Hutter, Alexander Szekely, Ilse Wimberger, Christian Pfaffel-Janser, and Andreas Neppach. Security Processor with

Quantum Key Distribution. *In Proceedings of the 19th International Conference on Application-Specific Systems, Architectures and Processors - ASAP 2008, Leuven, Belgium, 2-4 July, 2008, Proceedings*, volume 19, pages 37-42. IEEE Press, July 2008.

6. Michael Hutter, Alexander Szekely, and Johannes Wolkerstorfer. Embedded System Management using WBEM. *In Integrated Network Management - IM 2009, 11th IFIP/IEEE International Symposium, New York, USA, June 1-5, 2009, Proceedings*, pages 390-397. IEEE Press, June 2009.

7. Michael Hutter, Marcel Medwed, Daniel Hein, and Johannes Wolkerstorfer. Attacking ECDSA-Enabled RFID Devices. *In Applied Cryptography and Network Security - ACNS 2009, 7th International Conference, Paris-Rocquencourt, France, June 2-5, 2009, Proceedings*, volume 5536 of *Lecture Notes in Computer Science*, pages 519-534. Springer, June 2009.

8. Michael Hutter. RFID Authentication Protocols based on Elliptic Curves: A Top-Down Evaluation Survey. *In International Conference on Security and Cryptography - SECRYPT 2009, it is part of ICETE - The International Joint Conference on e-Business and Telecommunications, Milan, Italy, July 7-10, 2009, Proceedings*, pages 101-110. INSTICC Press, July 2009.

9. Thomas Plos, Michael Hutter, and Martin Feldhofer. On Comparing Side-Channel Preprocessing Techniques for Attacking RFID Devices. *In Workshop on Information Security Applications - WISA 2009, Busan, Korea, August 25-27, 2009, Proceedings*, volume 5932 of *Lecture Notes in Computer Science*, pages 163-177. Springer, August 2009.

10. Jörn-Marc Schmidt, Michael Hutter, and Thomas Plos. Optical Fault Attacks on AES: A Threat in Violet. *In the 6th Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2009, Lausanne, Switzerland, September 6, 2009*, pages 13-22. IEEE Press, 2009.

11. Michael Hutter, Thomas Plos, and Jörn-Marc Schmidt. Contact-Based Fault Injections and Power Analysis on RFID Tags. *In European Conference on Circuit Theory and Design - ECCTD 2009, 19th IEEE European Conference, Antalya, Turkey, August 23-27, 2009, Proceedings*, pages 409-412, IEEE Press, 2009.

12. Martin Feldhofer, Manfred Josef Aigner, Michael Hutter, Thomas Plos, Erich Wenger, and Thomas Baier. Semi-Passive RFID Development Platform for Implementing and Attacking Security Tags. *In Workshop on RFID / USN Security and Cryptography - RISC 2010, London, England, November 8-11, 2010, Proceedings*, IEEE Press, 2010.

13. Michael Hutter, Martin Feldhofer, and Thomas Plos. An ECDSA Processor for RFID Authentication. *In Workshop on RFID Security - RFIDsec*

2010, 6th Workshop, Istanbul, Turkey, June 7-9, 2010, Proceedings, volume 6370 of *Lecture Notes in Computer Science*, pages 189-202. Springer, November 2010.

14. Michael Hutter and Ronald Tögl. A Trusted Platform Module for Near Field Communication. *In Conference on Systems and Networks Communications - ICSNC 2010, 5th International Conference, Nice, France, August 22-27, 2010, Proceedings*, pages 136-141. IEEE Press, 2011.

15. Jörn-Marc Schmidt, Thomas Plos, Mario Kirschbaum, Michael Hutter, Marcel Medwed, and Christoph Herbst. Side-Channel Leakage Across Borders. *In Smart Card Research and Advanced Applications 9th IFIP WG 8.8/11.2 International Conference - CARDIS 2010, Passau, Austria, April 14-16*, volume 6035 of *Lecture Notes in Computer Science*, pages 189-202. Springer, November 2010.

16. Michael Hutter, Martin Feldhofer, Johannes Wolkerstorfer. A Cryptographic Processor for Low-Resource Devices: Canning ECDSA and AES like Sardines. *Workshop in Information Security Theory and Practice - WISTP 2011, Heraklion, Greece, June 1-3*, Springer, to appear.

Contributions to Refereed Workshops Without Formal Proceedings

1. Thomas Plos, Michael Hutter, Martin Feldhofer, Evaluation of Side-Channel Preprocessing Techniques on Cryptographic-Enabled HF and UHF RFID-Tag Prototypes. *In Workshop on RFID Security - RFIDSec 2008, Budapest, Hungary, July 9-11, 2008.*

2. Christoph Nagl and Michael Hutter. Coupon Recalculation for the Schnorr and GPS Identification Scheme: A Performance Evaluation. *In Workshop on RFID Security 2009 - RFIDSec 2009, 5th edition, Leuven, Belgium, June 30-July 2, 2009.*

Master's Thesis

1. Michael Hutter, EM Side-Channel Attacks on Cryptographic Devices. Master's thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, Februar 2007.

Glossary

DH	Diffie-Hellman
AES	Advanced Encryption Standard
ALU	Arithmetic Logic Unit
AMBA	Advanced Microcontroller Bus Architecture
ASIC	Application Specific Integrated Circuit
CMOS	Complementary Metal Oxide Semiconductor
CPA	Correlation Power Analysis
DEMA	Differential Electromagnetic Analysis
DER	Distinguished Encoding Rules
DES	Data Encryption Standard
DPA	Differential Power Analysis
DRP	Dual-Rail Precharge
DSA	Digital Signature Algorithm
DSO	Digital Sampling Oscilloscope
DSS	Digital Signature Standard
DST	Digital Signature Transponder
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EEPROM	Electrically Erasable Programmable Read-Only Memory
EM	Electromagnetic
FFT	Fast Fourier Transformation
FIB	Focused Ion Beam
FPGA	Field Programmable Gate Array
GE	Gate Equivalent
GPS	Girault-Poupard-Stern
HD	Hamming Distance
HF	High Frequency
HW	Hamming Weight

I/O	Input/Output
ISO	International Organization for Standardization
LMS	Least Mean Square
LSB	Least-Significant Bit
MDPL	Masked Dual-Rail Precharge Logic
MSB	Most-Significant Bit
NIST	National Institute of Standards and Technology
OBIC	Optical Beam Induced Current
PC	Personal Computer
PET	Polyethylene Terephthalate
PKI	Public-Key Infrastructure
PRNG	Pseudo-Random Number Generator
RAM	Random Access Memory
RF	Radio Frequency
RFID	Radio-Frequency Identification
RISC	Reduced Instruction Set Computer
RNG	Random Number Generator
ROM	Read Only Memory
RPC	Randomized Projective Coordinates
SABL	Sense Amplifier Based Logic
SASEBO	Side-Channel Attack Standard Evaluation Board
SCA	Side Channel Analysis
SHA	Secure Hash Algorithm
SNR	Signal-to-Noise Ratio
SPA	Simple Power Analysis
SRAM	Static Random Access Memory
UHF	Ultra-High Frequency
UID	Unique Identifier
UMC	United Microelectronics Corporation
VHDL	Very High Speed Integrated Circuit Hardware Description Language
WSB	Write-Single-Block

ZPA	Zero-value Point Attacks

1 Introduction

Radio Frequency Identification (RFID) is a wireless technology that gained large attention in the last decade. The major advantages of this technology are the simple ease of use due to a contactless operation and the digital information transfer between objects in a near proximity. RFID has found its way in many applications such as supply-chain management, tracking of animals and goods, cashless payment, or access control. This thesis focuses on security aspects of this technology and covers the secure and efficient implementation of such devices.

RFID devices basically consist of a tiny microchip that is connected to an antenna. These so-called tags are typically powered passively by a reader which generates an electromagnetic field. This field is also used as a communication channel between the reader and the tags. Due to the passive operation, implementations have to meet several requirements. Among the most important requirements is the low-power consumption. RFID tags have to consume only a few microwatts of power to operate also at a distance to the reader. Next to the low-power requirements, implementations are restricted by chip-area limits. This is because the larger the chip area, the higher will be the production costs in general which encourages low-area designs in order to allow a large-scale deployment of tags in the market.

Besides these requirements and the growing demand and integration of RFID devices also in security-related applications, there exist the stringent requirement of implementation security. It has shown in the past that electronic devices leak information through physical side channels that can be exploited in implementation attacks. There have been many contributions from the cryptographic community that demonstrate side-channel attacks on cryptography-enabled devices that are able to reveal the secret key using different attacking techniques. There have been also many proposals to extract secret information by inducing

faults into such devices. Side-channel attacks and also fault attacks pose one of the most powerful attacks nowadays that can be performed even with low costs. Hardware as well as software designers have to consider such attacks in practice and require to integrate appropriate countermeasures that makes such attacks more difficult to succeed.

This thesis focuses on secure and efficient implementations of passive RFID tags that support cryptography. It is structured into two main parts. In the first part, we evaluate the susceptibility of RFID devices against different implementation attacks. We contribute by identifying weaknesses of such devices that are commercially available and used in various applications nowadays. The second part focuses on a hardware implementation of a passive tag that implements countermeasures against these attacks. We improve the state-of-the-art in efficient implementation of Elliptic Curve Cryptography (ECC) on resource-constrained devices and propose a processor that meets the demanding requirements of passive RFID tags that are applied in security applications.

1.1 Our Contribution

Our contribution to the field of RFID security consists of several investigations. At the beginning of my research, there was no assurance if side channel attacks or fault attacks pose a serious threat to RFID devices or not. Since there have not been many RFID devices which support cryptographic functionalities, we started to design a passively powered RFID-tag prototype that implements AES in hardware as well as in software. The prototype allowed us to demonstrate the susceptibility to different implementation attacks. Together with Stefan Mangard and Martin Feldhofer, we first presented power and electromagnetic attacks on that platform in [HMF07]. The work identifies side-channel leakages of RFID devices that can be even exploited within the field of the reader. This findings emphasize the need of countermeasures against these kind of devices.

Since attacks on passively powered RFID devices need special measurement setups, we evaluated different methods to increase the performance of such attacks. In [PHF08], we investigated different side-channel pre-processing techniques such as filtering, trace decimation, and Differential Frequency Analysis (DFA) attacks. We also analyzed AES implementations with randomization countermeasures and present successful attacks on RFID-tag prototypes. The work was mainly done in collaboration with Thomas Plos and Martin Feldhofer. In [PHH08, PHF09], we further improved the measurements by a special low-cost shielding technique that reduces the impact of electromagnetic radiations in the proximity of side-channel measurements. This was a joint work with Thomas Plos and Christoph Herbst. For further analysis of RFID-based Application Specific Integrated Circuit (ASIC) designs, we developed an RFID-tag prototype that assembles a Field Programmable Gate Array (FPGA) in [FAH+10].

The threat of fault-analysis attacks on general-purpose devices has been investigated together with Jörn-Marc Schmidt in [SH07]. In this work, we performed optical and EM attacks on a Chinese Remainder Theorem (CRT)-based

RSA implementation. We have been able to flip individual bits of SRAM cells and affect the program flow of a microcontroller implementation. Furthermore, we introduced a new non-volatile fault-attack using high-voltage spark gaps. The impact of ultra-violet light on an AES implementation has been evaluated in [SHP09] together with Jörn-Marc Schmidt and Thomas Plos.

First results of fault-analysis attacks on passive RFID devices have been reported in [HSP08]. Together with Thomas Plos and Jörn-Marc Schmidt, we introduced several fault-analysis setups and targeted the writing of data into the memory of commercially-available RFID devices. The results of our outcomes are also described in Chapter 3 of this thesis. In [HSP09], we performed spike and glitch attacks on RFID devices by presenting a new setup to perform fault attacks on RFID. Furthermore, we present a contact-based measurement setup to perform power-measurements of RFID tags.

After evaluating RFID devices that implement no or at least symmetric cryptographic functionalities, we analyzed RFID devices that support public-key cryptography. Different elliptic-curve based public-key protocols that are suitable for RFID devices have been analyzed in [Hut09]. First results of power and electromagnetic attacks on an RFID-based ECDSA hardware implementation has been published together with Marcel Medwed, Daniel Hein, and Johannes Wolkerstorfer in [HMHW09]. Furthermore, we compared the elliptic-curve based protocol of Schnorr and GPS in [NH09]. The latter one has been a joint work together with Christoph Nagl.

In the Austrian government funded project *CRYPTA*, we targeted a low-resource implementation of both AES and ECDSA for passive RFID devices. This project has been a joint work with my colleagues Martin Feldhofer, Johannes Wolkerstorfer, Thomas Plos, and the semiconductor manufacturer austriamicrosystems (AMS). We made use of the findings obtained in the previous described contributions in order to design a secure and efficient RFID tag supporting symmetric as well as asymmetric cryptography. The requirements for the processor are low area, low power, appropriate speed, and a sufficient level of security. We made several contributions regarding implementation optimizations. First, we optimized the hardware processor on an algorithmic level. For this, we proposed a new technique by sharing a common-Z coordinate in elliptic curve cryptography (ECC). This work has been done together with Marc Joye (Technicolor) and Yannick Sierra (Oberthur Technologies). Second, we made optimizations on the arithmetic level and implementation level. A new approach has been proposed that uses a microcode-control architecture with a tiny 8-bit microcontroller in order to reduce the area requirements of our processor due to sharing of resources and to provide high speed for ECC operations. Finally, we integrated several countermeasures against implementation attacks. Implementation details are given in [HFP10, FAH+10, HFW11] and in Chapter 6 of this thesis.

1.2 Structure of the Thesis

The thesis is structured into two main parts. In the first part, we investigate the susceptibility of RFID devices against implementation attacks. An introduction into side-channel attacks and the performed analyses are given in Chapter 2. Chapter 3 presents results of performed fault attacks on RFID devices. Attacks on public-key enabled RFID devices are discussed in Chapter 4. The second part of the thesis focuses on low-resource implementations for RFID devices that provide countermeasures against the described attacks. Chapter 5 presents new formulæ for elliptic-curve cryptography implementations that reduce the memory requirement to a minimum while providing very fast and efficient computations. Finally, we apply these formulæ in a hardware implementation that supports ECDSA as well as AES described in Chapter 6. Conclusions are drawn in Chapter 7.

In the following, we briefly describe the outline of each chapter in a few sentences:

Chapter 2 introduces to side-channel attacks on RFID devices. First, we will describe side-channel attacks in general and apply these attacks on passively powered RFID tags in particular. After giving related work on that topic, we performed power-analysis attacks on RFID tags by measuring the power consumption in the antenna connections and by probing chip internal power-supply lines. Next, we performed different electromagnetic attacks using various setups and analyze the timing behavior of commercial available tags. Results are given for each setup that identify weaknesses of commercially available RFID tags due to the side-channel leakage of information. Finally, we discuss common countermeasures to thwart these attacks.

Chapter 3 evaluates fault-analysis attacks on RFID devices. After a general introduction into fault attacks, we present four different setups to inject faults by spike and glitches, electromagnetic interferences, optical inductions, and by tearing the antenna of passive RFID devices. We targeted the writing of data into the memory of such devices and show how this can be prevented by the injection of faults. In addition, we show how to modify the content of the memory without being detected. Like in Chapter 2, we discuss common countermeasures to thwart these attacks.

Chapter 4 presents side-channel attack results performed on public-key enabled RFID devices. After an introduction into public-key cryptography, we give related work and describe our attack in detail. In fact, we targeted an ECDSA implementation in hardware and performed power and electromagnetic attacks on a passive RFID-tag prototyping platform. All performed attacks have been successful and revealed the private key. The given attack allow adversaries to forge digital signatures and thus impersonating ECDSA-enabled RFID tags. Finally, we summaries our outcomes and give conclusions.

Chapter 5 provides new formulæ for elliptic-curve cryptography over prime fields. The formulæ allow very efficient computations while reducing the memory requirements to a minimum. In fact, we applied a new technique where each point during scalar multiplication share a common Z-coordinate. Furthermore, we first introduce *out-of-place* operations for ECC that guarantee that no additional memory is needed to perform multiple-precision arithmetic computations. The gain and the impact on the memory requirements are given in the result section. Conclusions are drawn afterwards.

Chapter 6 applies the obtained formulæ of Chapter 5 in a hardware implementation of ECDSA and AES. In particular, we present a low-resource hardware processor over $\mathbb{F}_{p192}$ that targets low-resource devices like RFID tags. Instead of a finite-state machine based approach, we followed a new approach by implementing a tiny 8-bit microcontroller with a microcode-control engine which increases the flexibility and scalability of our implementation to a maximum. Moreover, we applied several countermeasures against side-channel and fault attacks that have been presented in Chapter 2, Chapter 3, and Chapter 4.

Conclusions of the thesis are drawn in **Chapter 7**.

Part I

Implementation Attacks

2

Side-Channel Attacks on RFID Devices

It's leaking from...everywhere
— Adam Savage (MythBusters) after the test dummy's legs start leaking gel.

In this chapter, we introduce side-channel attacks that have been performed on passively-powered Radio-Frequency Identification (RFID) devices. First, we will describe side-channel attacks in general and discuss appropriate measurement setups to perform these attacks on passive RFID devices. We will present setups for power-consumption measurements, sensing of electromagnetic tag emanations, antenna tearings, and analyses of the timing behavior. Furthermore, we will present results of performed attacks on an implementation of the Advanced Encryption Standard (AES)[Nat01] that runs on different RFID-tag prototypes or the writing of data in the memory of commercially available RFID tags[1]. Our results show that passive RFID devices are susceptible to all of these attacks which emphasizes the need of appropriate countermeasures.

The main contribution of this chapter is to highlight the vulnerability of RFID devices to side-channel attacks. We show that security-related implementations need appropriate countermeasures to thwart these attacks. Without countermeasure implementations, an adversary would be able to extract secret information from that RFID device. This can be the secret key of a cryptographic-enabled RFID tag which would allow the cloning of tags and thus the impersonation of RFID-tagged entities like counterfeited goods.

Note that this chapter is a summary of several contributions together with Stefan Mangard, Martin Feldhofer, Thomas Plos, Christoph Herbst, Marcel Medwed, Daniel Hein, Johannes Wolkerstorfer, and Jörn Marc-Schmidt. Parts

[1] Most of the analyzed tags do not implement any cryptographic operations.

of the work have been published at the International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007) [HMF07], International Workshop on RFID Security (RFIDsec 2008) [PHF08], National Workshop on Microelectronics (Austrochip 2008) [PHH08], International Workshop on Information Security Applications (WISA 2009) [PHF09], European Conference on Circuit Theory and Design (ECCTD 2009) [HSP09], the International Conference on Applied Cryptography and Network Security (ACNS 2009) [HMHW09], and the International Journal of Security and Networks (IJSN 2010) [HPF10].

2.1 An Introduction to Side-Channel Attacks

Side-channel analyses are related to implementation attacks that work passively by measuring physical properties of a cryptographic device. The attacks exploit the fact that the physical properties are dependent on the data and the operations that are processed by the device. Typical physical properties of electronic devices are the power consumption, the electromagnetic emanation (EM), the execution time, or the temperature variation.

The first side-channel analysis results have been reported by P. Kocher in 1996 [Koc96]. He exploited the execution time of cryptographic algorithms that were running on a standard PC. He showed how to extract secret information by using only timing information of the implemented algorithm. Three years later, he and his colleagues J. Jaffe and B. Jun [KJJ99] introduced power-analysis attacks. Power-analysis techniques make use of the fact that the power consumption of a cryptographic device depends on the information it processes. The same relation holds for electromagnetic emissions of a device which has been first observed by K. Gandolfi et al. [GMO01] and D. Agrawal et al. [AARR03].

Besides other existing side channels, the power consumption and electromagnetic emanation have gained most importance because they are easily exploitable by appropriate measurement setups. The power consumption can be measured by inserting a resistor to the power-supply lines of the device. The voltage drop can be measured using a digital storage oscilloscope. Electromagnetic emanation, in contrast, can be measured similarly using an electromagnetic probe. The probe is placed on top of the cryptographic device and senses information that can be gathered by an oscilloscope. There exist various types of EM probes that can be used to measure magnetic or electric near-field signals or electromagnetic far-field emanations [Man03].

Power analysis in its simplest form requires only a single power trace that is acquired during the execution of the cryptographic algorithm for deducing the secret of the device. This technique is called Simple Power Analysis (SPA) and can be conducted, for example, by visually inspecting the power trace. Differential power analysis makes use of statistical methods to extract data-dependent information. There, the information of multiple power traces is combined to reveal the secret key used by the device. We refer to DPA when the power consumption of a device is exploited as a side-channel source. We refer to Differential Electromagnetic Analysis (DEMA) when the electromagnetic emanation is

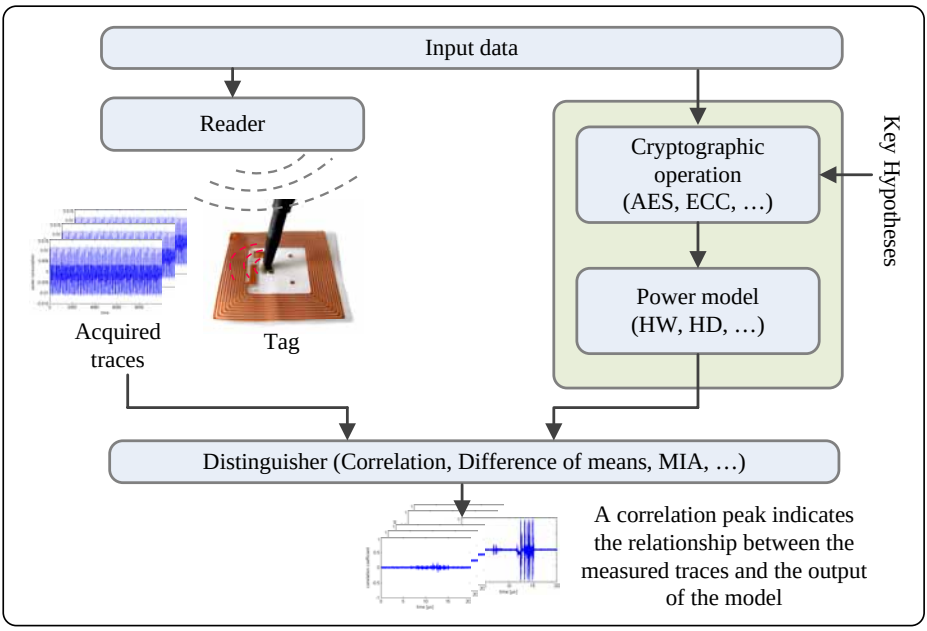


Figure 2.1: Side-channel analysis methodology. Next to the measurement of traces, a model is used to characterize the power consumption for the processing of a specific intermediate value. Statistical methods are used to compare the outcome of the model with the acquired side-channel traces.

exploited.

A DPA attack works as shown in Figure 2.1. We target an RFID tag that performs some cryptographic operation. First, the tag is fed with input data from the reader which is transmitted over the air interface. The tag performs some computation with the data and sends the response back to the reader. During the computation, a measurement probe is placed on top of the chip die. The probe is connected to an oscilloscope which measures the acquired side-channel (electromagnetic, power, etc.) traces. Next to the physical measurement of side-channel information, a model is constructed in software that predicts the consumed power of the cryptographic device. We used Matlab [The] in all of our experiments. The model predicts intermediate values of the cryptographic algorithm by deploying the known input data and hypothetical values for the unknown secret key of the device. After that, these predicted intermediate values are fed into a power-consumption model which results in predicted power-consumption traces. For this power model, the Hamming weight (HW) or the Hamming distance (HD) of the intermediate values is usually used which often provides a good approximation of the real power consumption of the device. In the last step of the attack, the predicted power-consumption traces are compared to the physically measured traces by means of statistical methods (so-called distinguisher). Typically used distinguishers are the Difference-of-Means

(DoM) where the difference of two sets of power traces is calculated to identify their relation to each other. Another often used distinguisher is the Pearson correlation-coefficient that calculates the linear dependency between a vector of power-consumption samples x and predictions y. A DPA attack is successful if the predicted power consumption, which is related to the correctly guessed secret key, shows a significant linear dependency with the measured power-consumption samples which causes a peak in the resulting correlation trace. The correlation function is defined by

$$\rho = \frac{\sum_{i=1}^{n} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \cdot \sum_{i=1}^{n} (y_i - \bar{y})^2}}, \tag{2.1}$$

where n denotes the number of measured power traces. A DPA attack that applies the correlation method is therefore often referred as Correlation Power Analysis (CPA) according to E. Brier et al. [BCO04]. Other distinguisher are Mutual Information [GBTP08], Bayesian attacks (used in e.g. template attacks) [CRR03], or the stoachastic model of W. Schindler [SLP05]. In the following chapters, we apply the Pearson's correlation coefficient as a distinguisher for DPA attacks. A more detailed description of power-analysis attacks can be found in [MOP07].

2.1.1 Side-Channel Analysis on Passive RFID Devices

In the following, we give an overview of various issues regarding the acquisition and analysis of side-channel information of passive RFID devices.

As opposed to active devices, passive devices get supplied by an external power source. In view of RFID, readers provide an electromagnetic field that power the devices within their reading range. These passive devices (tags) mainly consist of a microchip that is connected to an antenna. The microchip has only two antenna connections that provide the tag with power, a clock signal, and communication data. Contact-based power measurements on passive RFID devices are therefore only possible by inserting a resistor either into the antenna lines or into internal power-supply lines of the microchip. The latter method requires a sophisticated measurement setup like a probing station to get contact with the internal power-supply lines.

A more convenient way to gather side-channel information of passive RFID devices is to exploit the electromagnetic emanation of the microchip. The current flow within the microchip produces an electromagnetic field. This field contains different signals such as the square-wave clock or signals that are caused by data processing. These signals can be sensed by magnetic near-field probes that are placed directly on the surface of the chip [AARR03]. However, the emanated signals of passively powered devices are very weak so that special measurement equipment is necessary to amplify the side-channel information. In fact, passively powered devices consume only few microwatts of power whereas actively powered devices typically consume some milliwatts. This is a factor of thousand which makes measurements on such devices harder to perform.

Next to the very weak side-channel signal, there exist a high noise source in RFID-based measurement setups. Passive tags get powered by the electromagnetic field of a reader which is typically between 40 dB and 80 dB higher than the signals emitted by the tags. As a result of side-channel measurements, interesting signal emissions of the tags may be unintentionally overwhelmed by the occurring interferences of the strong reader field. The data-acquisition resolution of the measurement equipment is thus inevitably reduced since the weak signals of the tag are superimposed onto the reader field. In addition to the lower acquisition resolution, side-channel traces get misaligned in both the time and the amplitude dimension since the reader field often becomes de-synchronized due to the modulation of the field. In fact, the reader is a high noise source and makes side-channel measurements on RFID tags difficult to perform. The main challenge in that context is therefore to minimize the impact of the reader field and to overcome the resulting misalignment of measured side-channel traces.

We applied several pre-processing techniques in our experiments. First, we used different filters described in the following to remove the carrier of the reader signal from the measured traces. Second, we calculated the envelope signal of the traces in most cases which reduces the noise due to misalignments. This is done by taking the absolute values of the traces and by applying a low-pass filter afterwards. Finally, we aligned all traces in both vertical and horizontal orientation.

2.1.2 Related Work

There exist several attacks on RFID devices. One of the first attacks has been presented by Bono et al. [BGS+05] in 2005. They performed an attack on the Digital Signature Transponder (DST) from Texas Instruments, which is a passively powered 135 kHz RFID transponder used for car-immobilizer systems (in fact about 150 million DST transponders have been shipped worldwide) and cashless-payment applications (e.g. the Exxon-Mobile Speedpass). The transponder implements a 40-bit symmetric block cipher to provide a challenge-response authentication. For the attack, the authors used 16 Field Programmable Gate Array (FPGA) units to perform a brute-force attack on the secret key. They were able to recover the key in less than one hour.

In 2007, K. Nohl and H. Plötz [Noh08, NESP08] and G. de Koning Gans et al. [dKGHG08] presented an attack on the Mifare Classic RFID system. Mifare Classic is a product from NXP and has been applied in about 200 millions of applications worldwide such as contactless payment, ticketing, and access control (i.e. about 70 % of the total contactless smart-card market). As a cryptographic primitive, it uses the proprietary *Crypto-1* cipher that is used to authenticate the transponder with a challenge-response protocol. The authors analyzed the chip by reverse engineering and revealed the algorithm of *Crypto-1*. During their work, they discovered several security flaws in the system which allows several algebraic attacks to be performed on Mifare Classic. F. Garcia et al. [GdKGM+08] presented an algebraic attack on Mifare Classic in 2008. Their attack needs less than one second to recover the secret key. They improved the attack in

2009 [GvRVS09] and demonstrated card-only attacks on Mifare Classic which is in fact the most realistic attack scenario [Cou09].

The first side-channel attack on RFID devices has been presented by Y. Oren and A. Shamir [OS06, OS07] in 2006. They showed that it is able to reveal the kill password of ultra-high frequency (UHF) tags by simply observing single power traces that are reflected from the tag to the reader. This backscattered power trace changes depending on the processed data.

In 2007, we first presented results of a DPA attack on passive RFID devices [HMF07]. We analyzed hardware and software AES implementations of high frequency (HF) tag prototypes by means of power and electromagnetic analysis. All devices have been successfully attacked using less than 1 000 power traces. T. Plos [Plo08] demonstrated the susceptibility of UHF tags against DPA attacks in 2008. He analyzed commercially-available RFID tags and determined data-dependent emanations at a distance of up to one meter.

In 2008, T. Eisenbarth et al. [EKM+08] presented the first power-analysis attack on a commercially available RFID system. They performed an attack on the proprietary *Keeloq* Code Hopping Scheme. *Keelog* is a block cipher that has been used in many remote keyless entry systems. It uses two different keys: a manufacturer key and a device key of 64 bits. Encryption and decryption operations are performed on blocks of 32 bits. As an outcome of the proposed attack, they showed to reveal the manufacturer key from a receiver using less than 1 000 power traces and recovered the device key of the remote control with less than 10 traces. Having recovered this key, they have been able to clone transponders to open existing systems such as buildings as well as car and garage doors which apply *Keeloq* as a cryptographic primitive.

2.2 Analysis of the Power Consumption

In order to perform power analysis on RFID devices, corresponding measurement setups are necessary. These setups are typically more complex than power-measurement setups for contact-based devices. The reason for this is that RFID devices are powered via the electromagnetic field of an RFID reader and not directly via a power-supply unit. The classical method to measure the power consumption of a device is to insert a resistor into one of its power-supply lines. The voltage drop over this resistor, which is proportional to the consumed power of the device, can then be measured by a digital storage oscilloscope [MOP07].

In practice, the insertion of a resistor into the power-supply line of an RFID device is often not possible. As already stated in Section 2.1.1, RFID devices are usually integrated on a single die providing only two connections for the antenna. Thus, there are only two ways to measure the power consumption of RFID devices. One way is to insert a resistor into the antenna lines of the device. This method is destructive because a resistor has to be soldered between the chip and the antenna connections. The other way is to use a probing station to place a resistor between the cryptographic unit and the analog front-end of the RFID device. We performed power-analysis attacks using both methods and

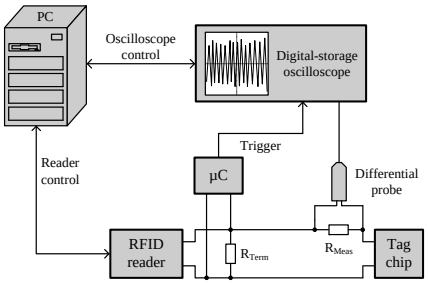


Figure 2.2: Schematic view of the DPA measurement setup.

Figure 2.3: Setup for power-consumption measurements on a passive RFID tag.

describe them in the following.

2.2.1 Measurements at the Antenna Connections

We performed a power-analysis attack on a commercially available HF RFID tag that operates at 13.56 MHz. The tag is ISO/IEC 15693 [Int01b] standard compliant but does not support any cryptographic operations. It only provides commands to write and read from the memory. Hence, we decided to target the writing of data in the memory of the tag to demonstrate that data which is processed by the tag can be extracted by side-channel analysis. We assume that for the attack it does not matter which data is processed and if the targeted values are intermediates of cryptographic operations or simple commands such as reading or writing.

Figure 2.2 shows the schematic view of our setup. It is composed of the following components: a PC, an ISO/IEC 15693 compatible RFID reader, a commercially available 13.56 MHz HF RFID tag, a microcontroller with an analog front-end, a digital-storage oscilloscope, a differential probe, an impedance-matching resistor, and a measurement resistor.

The RFID reader is connected to the PC over a serial connection. Instead of an antenna, we connected a 50 Ohm impedance-matching resistor R_{Term} that is used to match the impedance of the reader antenna output. The tag has been connected in parallel to the matching resistor over a two wire cable. For this, we separated the chip from its antenna and the antenna pads of the tag have been soldered on a two-pin strip (shown in Figure 2.3). Next to that, we connected a self-made trigger device to the antenna output. The trigger device is used to detect the sending of data of the reader to the RFID tag. It mainly consists of a fast envelope detector that is used to transform the analog signal of the reader into digital signals. The trigger device is then connected to an 8-bit ATmega128 [Atm07] microcontroller from Atmel. It provides 128 kB of Flash memory and allows easy interaction with other components by providing 53

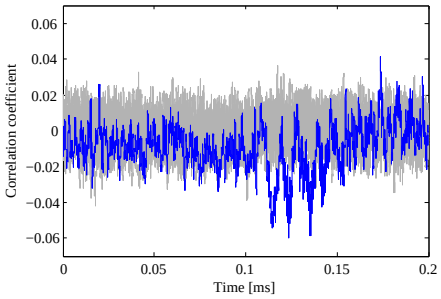


Figure 2.4: Result of a DPA attack on a commercially HF tag using traces from the antenna connections.

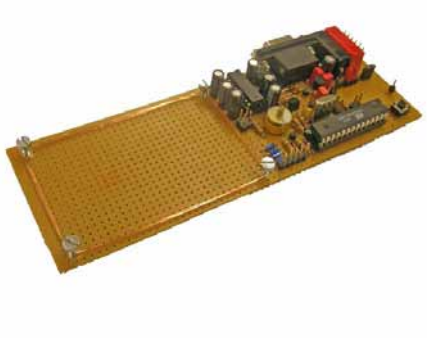


Figure 2.5: A passively powered RFID-tag prototype operating at a frequency of 13.56 MHz.

customizable I/O pins. The microcontroller was programmed to set a dedicated output pin to high as soon as a write command is received. This signal can then be used to trigger the storage oscilloscope to measure the power consumption of the tag. For this, we used a 100 Ohm measurement resistor which is placed in series to the matching resistor and the RFID chip. The consumed power of the tag is then measured over this resistor using an active 1 GHz differential probe (AP034 from LeCroy). The voltage drop across the inputs of the differential probe is captured by an 8-bit digital oscilloscope which has a 1 GHz bandwidth (LC584AM from LeCroy). Figure 2.3 shows a picture of the RFID chip and the measurement resistor. A detailed description of the setup is given in [HSP09].

We performed a correlation power-analysis attack on that RFID tag by measuring 11 000 traces with a sampling rate of 100 MS/s. The target of the attack has been an 8-bit value that has been written into the internal EEPROM memory of the tag. The power consumption during the time when the value is processed and stored in the memory was measured using the oscilloscope. We used the Hamming-weight power model to characterize the power consumption of the tag.

In Figure 2.4, the result of the attack is shown. The x-axis represents the acquisition time of 200 µs. The y-axis shows the result of the attack as an output of the Pearson correlation. The correlation trace of the correct hypothesis (i.e. the correct value that has been written in the memory) is drawn in blue (or black in black/white prints) while all other incorrect hypotheses are drawn in gray. The highest absolute correlation is about 0.06 and shows a significant peak between 100 and 150 microseconds. The performed attack has been successful and revealed the data that was written in the memory of the tag.

2.2.2 Probing Internal Power-Supply Lines

Another possibility to measure the power consumption of RFID tags is the insertion of a measurement resistor to the internal power-supply lines of an RFID

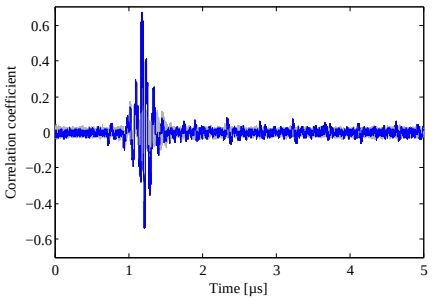


Figure 2.6: Result of a DPA attack on a passively powered RFID-tag prototype.

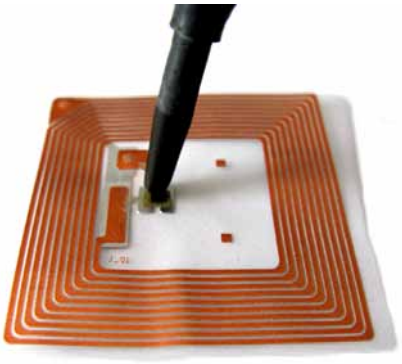


Figure 2.7: A magnetic near-field probe on top of the surface of an HF RFID tag.

chip. One way to achieve this measurement is to use a probing station. The probing station allows the monitoring of the power consumption of the tag by penetrating a tiny needle through the top-level substrate of the chip. The power consumption can then be easily acquired by an oscilloscope.

For this experiment, we used an RFID-tag prototype to demonstrate the feasibility of such an attack. Figure 2.5 shows a picture of the prototype. It mainly consists of an antenna, an analog front-end, and a microcontroller. The antenna consists of four windings and has a shape of contactless smart cards according to ISO 7810 [Int03]. It is connected to the analog front-end of the prototype which is responsible to demodulate and modulate the RF signals of the reader and the microcontroller. Furthermore, it is responsible to extract the power supply from the electromagnetic field of the reader. As a controller, we used the ATmega168 microcontroller from Atmel. The clock was provided by an assembled 13.56 MHz crystal oscillator which was synchronized with the field of the reader. All components are designed for low power consumption to make sure that the device can be powered passively by the field of an RFID reader. The perforated board consumes 3.6 mA at a minimum voltage of 1.75 V.

For the attack, we placed a measurement resistor between the analog front-end and the microcontroller of the prototype. As a target algorithm, we implemented AES in software that runs on the microcontroller. A challenge-response authentication protocol was used as proposed by M. Feldhofer [Fel04] that encrypts a challenge from the reader and responses with the ciphertext.

Figure 2.6 shows the result of a correlation power-analysis attack on the passively powered RFID-tag prototype. The target of the attack has been the first S-box output of the first round of AES. 10 000 power traces have been measured and the Hamming-weight power model has been used. The correct hypothesis (drawn in blue) for the first AES key byte led to a correlation coefficient of 0.67. All other hypotheses (drawn in gray) led to no significant correlation. A detailed description of the setup and attack is given in [HMF07].

2.3 Analysis of the Electromagnetic Emanation

Another approach to measure the power consumption of RFID devices is to measure it indirectly via the electromagnetic field. This can be done by a magnetic near-field probe shown in Figure 2.7. For this, the probe is placed on top of the chip surface. The current flow within the chip produces an electromagnetic field that is sensed by the probe. This field contains different signals such as the square-wave clock or signals that are caused by data-dependent processing. There exist several publications that discuss how this method can be used to attack contact-based devices (see for example [AARR03], [GMO01], and [QS01]).

Measurements of electromagnetic emanations of RFID tags typically contain signals of the strong reader field. This is because the tag and the probe has to be placed within the range of the reader during an attack. Since this field provides a strong noise source in side-channel measurements, it is necessary to minimize the impact of the reader signal as much as possible. There exist several ways to do this. One method is to use a special measurement bridge which cancels out the reader carrier during the side-channel measurements. This can be done by applying the so-called Helmholtz arrangement. Another solution would be to separate the chip from its antenna and to perform the measurements outside the reader field. However, this implies the destruction of the RFID inlay. As a third solution, one can filter specific frequency bands which contain data-dependent signals emitted by the device under attack. Special filters are therefore necessary. Finally, one can subtract the phase-shifted reader signal. All approaches increase the overall signal-to-noise ratio in RFID side-channel measurements and are discussed in the following.

2.3.1 Reader-Field Cancelation using the Helmholtz Arrangement

One way to overcome the problem of the interfering reader signal is the use of the test setup described in ISO/IEC 10373-6 [Int01a]. In this standard, a so-called Helmholtz assembly is specified for compliance testing of RFID tags. A picture of the assembly is shown in Figure 2.8. This setup essentially consists of two sense coils that are arranged in parallel to a reader antenna (the coil in the middle). The two sense coils are connected with in-phase opposition. Consequently, the induced voltage becomes zero at the point where the two coils are connected to each other (differential point). When a passively powered device gets close to one of these sense coils, it draws energy out of the field. The measuring bridge becomes unbalanced and an offset voltage can be observed at the differential point. This offset voltage is proportional to the power consumption of the RFID tag and can be used as a side-channel source. By measuring the voltage using a digital oscilloscope, information about the processed data of the tag can be revealed. Using such a setup reduces the carrier signal of an RFID reader by about 40 dB.

We performed a DEMA attack on our prototyping platform (see Figure 2.5) using the Helmholtz arrangement. We placed the tag near to one sense coil and

Figure 2.8: The Helmholtz Arrangement according to ISO/IEC 10373-6 [Int01a].

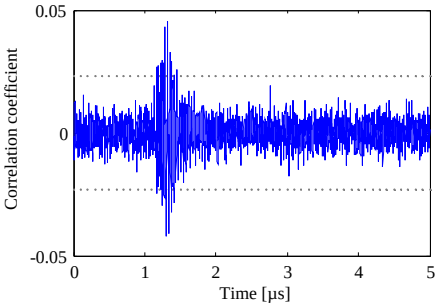


Figure 2.9: Result of a DEMA attack on our tag prototype using the Helmholtz arrangement.

measured the power-consumption at the differential point of the assembly. The target of our attack has been the first S-box output byte of the first round of AES. 30 000 traces have been acquired and the Hamming weight has been used as a power-consumption model. Figure 2.9 shows the result of the attack. The correct key hypothesis led to a correlation peak of about 0.048 after about 1.3 µs of the sampling window. Incorrect hypotheses showed no significant correlation.

The attack has been successful and revealed the secret key. These results are important because they demonstrate the feasibility to extract side-channel information of RFID tags even at a distance. Note that the distance of the sense coils to the reader antenna are specified to be 37.5 mm for ISO/IEC 10373-6 [Int01a]. However, our outcomes show that the data processing of RFID tags affect the reader field. This field can be in fact exploited by side-channel attacks. There have been several authors who already demonstrated that RFID data can be eavesdropped over the reader field not only in the standard-specified proximity of a reader (i.e. about 10 cm) but also in the far-field at a distance of up to several meters [Han08, uHK04, Rob06].

2.3.2 Antenna-Chip Separation

Another way to avoid the interfering reader field is to separate the RFID chip from its antenna. The basic idea of this method is to place an antenna in the field of a reader and to use wires to supply an RFID device that is placed outside the field. This approach has been first presented in [CLP05]. They separated the chip of a contactless smart card and connected the antenna using a cable. For this, they dissolved the card with Trichloroethylene at a temperature of 100 °C. The chip has been placed outside the reader field but gets powered by the separated antenna that lies within the field of a reader.

In Figure 2.10, a similar setup is shown where the chip of an RFID tag has been separated from its antenna. Two cable clamps have been used to establish a connection between the antenna pads of the chip and the separated antenna.

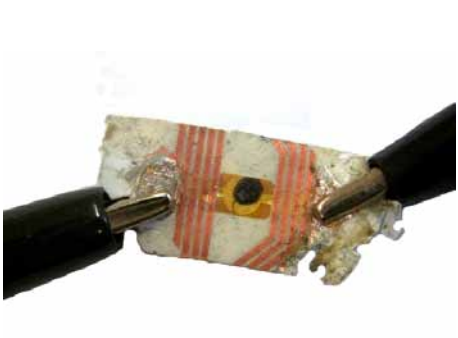


Figure 2.10: Chip separation from its antenna and placement outside the reader field.

Figure 2.11: EM shielding box used to increase the signal-to-noise ratio of SCA measurements.

This setup has been used in our experiments to perform electromagnetic attacks on RFID devices in order to reduce the impact of the reader field. The side-channel measurements can be further enhanced by performing the measurements within a shielded environment. We described such a setup in [PHH08] where we used a low-cost EM shielded box (about 300 €). A picture of the box is given in Figure 2.11. The RFID chip and the EM probe can be placed inside this shielded box whereas the reader is placed outside the box. It shows that such a setup can suppress unwanted signals up to several GHz which increases the signal-to-noise ratio through reduction of the noise.

2.3.3 Frequency-Selective Filtering using Receivers

An alternative method to minimize the impact of the reader field is to filter the signal that is measured by EM probes. This can be done by a receiver. Figure 2.12 shows two electromagnetic near-field probes and a test receiver from Rhode&Schwarz [Sch] to filter interesting side-band signals of RFID tags.

The first publication that discusses the use of a receiver to perform EM attacks on cryptographic devices is [AARR03]. In this publication, a receiver is used to find the side-channel leakage of devices in different parts of the EM spectrum. A very similar approach can of course also be used for RFID devices. In this case, the basic idea is to scan the EM spectrum of the device between the harmonics of the 13.56 MHz carrier. The leakage between the harmonics can easily be exploited as there is no interference of the carrier signal of the reader. Figure 2.13 shows the EM spectrum between 0 and 100 MHz where a 3 MHz filter has been used to identify data-dependent frequency bands.

Figure 2.12: Setup used to perform DEMA attacks by frequency-selective filtering using a receiver.

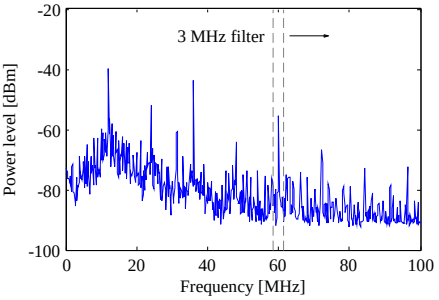


Figure 2.13: A receiver was programmed to sweep across the EM spectrum to identify data-dependent frequency bands.

In Figure 2.14, the result of an electromagnetic-analysis attack on our RFID-tag prototype is shown. The attack is split into two phases: a profiling phase and an attacking phase. In the profiling phase, the device was first separated from the reader field and characterized afterwards in terms of its side-channel leakage. We connected a 3 GHz wideband receiver (ESPI Test Receiver from Rhode&Schwarz [Sch]) to a magnetic near-field probe. The receiver was programmed to sweep across the EM spectrum of the tag prototype while it performed AES encryptions for different plaintexts. The sweep was performed using a filter bandwidth of 3 MHz. We have used the receiver for mixing down the HF signals to an intermediate frequency of 20.4 MHz which was sampled with the digital oscilloscope. We recorded 1 000 traces for different plaintexts for each 3 MHz interval between 10 MHz and 100 MHz. Subsequently, an electromagnetic-analysis attack was performed for each of the 30 frequency intervals. The upper plot in Figure 2.14 shows the EM spectrum from 0 to 100 MHz. The clock harmonics, which are a multiple of 13.56 MHz, can be clearly identified as peaks in the spectrum. The lower plot shows the correlation coefficient at each frequency interval. It can be observed that there are high data-dependent emissions in the sidebands of several clock harmonics. The highest obtained correlation coefficient is 0.39 at a frequency of 49 MHz.

After the profiling phase, we conducted an electromagnetic-analysis attack in presence of the reader field. In this phase, the receiver has been used as a simple filter that separates the highest data-dependent frequency interval from the rest of the interfering signals. Thus, we adjusted the receiver to 49 MHz and performed an attack using the 3 MHz-filtered sideband signals of the tag prototype. Using this method, each secret key byte of our AES implementation could be successfully revealed.

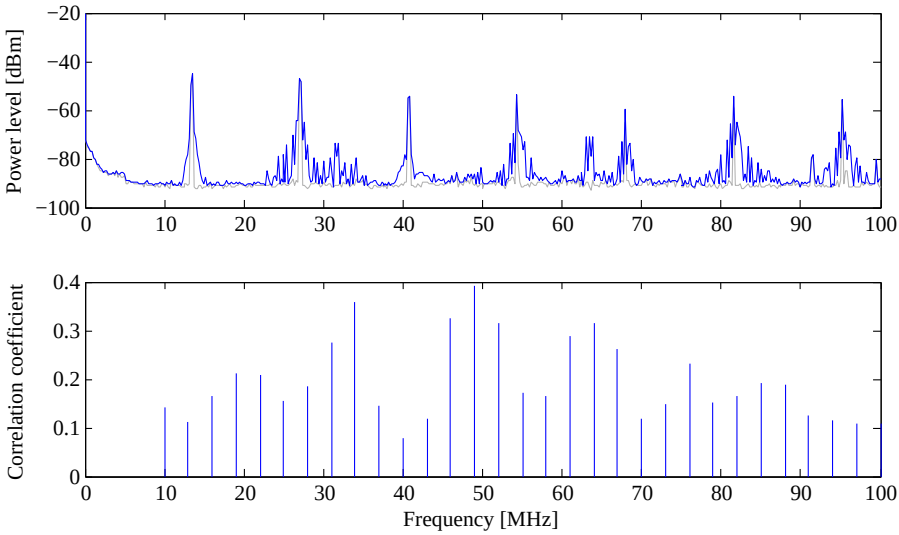


Figure 2.14: Correlation across the EM spectrum of the passively powered RFID-tag prototype.

2.3.4 Phase-Shifted Signal Subtraction

Another method to reduce the impact of the reader field is to use an appropriate measurement setup which subtracts the reader signal from the measurement traces. The first who used such a setup are T. Kasper et al. [KOP09]. Their setup is composed of an amplifier, a phase-shift circuit, and a signal subtraction unit. They used the 13.56 MHz clock output of an RFID reader and shifted the signal in phase before subtracting it from the acquired side-channel traces. After power-trace measurements, they successfully performed a DEMA attack on a contactless smart-card and revealed parts of the secret key.

2.4 Analysis of the Timing Behavior

The timing behavior of cryptographic devices provide one of the most powerful side channels. P. Kocher [Koc96] has been the first who exploited the runtime of different algorithms in 1996. He showed how to extract secret information out of cryptographic primitives such as RSA, Diffie-Hellman, and the Digital Signature Standard (DSS). The runtime of the analyzed algorithms vary in time depending on the data the device processes. Hence, an attack can be mounted that reveals the secret key used during the computation.

We observed the same behavior in several RFID tags nowadays. Figure 2.15 shows the time variation during the writing of different data into the EEPROM of an RFID tag. It shows that the power consumption raises earlier for certain intermediate values. These values are drawn in blue-scale colors and highlight

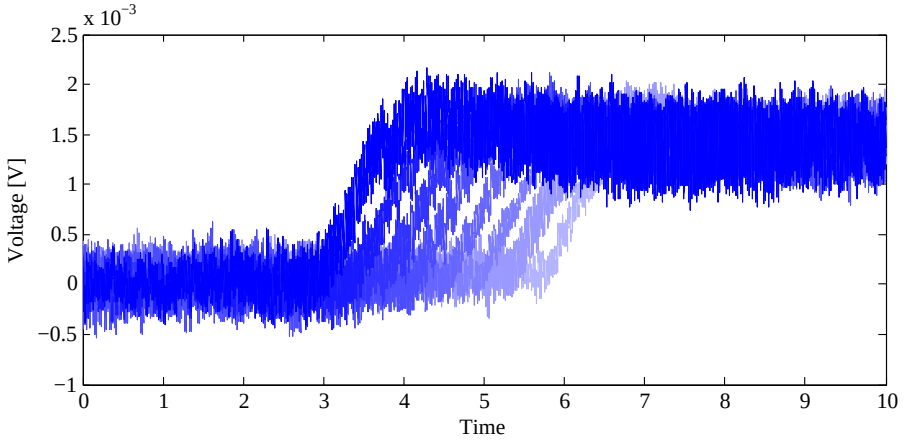


Figure 2.15: Data-dependent time variations in the execution time of an EEPROM write operation of an RFID tag.

the different timings when the chip pulls energy from the field to write the data in the memory. Depending on the value, it raises sooner or later. This time-depended information leakage can be used in side-channel attacks to extract the intermediate value that is processed by the tag.

We exploited the timing behavior of these tags in our experiments and constructed an appropriate power-consumption model. In fact, most attacks can be performed by simply applying the Hamming weight (HW) or Hamming distance (HD) power model since the devices usually leak the information according to the number of bits of the processed intermediates. However, the performance of our attacks could be significantly improved by exploiting the timing behavior in our power-consumption model.

2.5 Countermeasures against Side-Channel Attacks

Power and EM analysis attacks work because the power consumption of the target depends on the intermediate values of the cryptographic algorithm. The two best known principles of SCA countermeasures are *masking* and *hiding* [MOP07].

Masking tries to randomize the intermediate values that are processed on the device. On the architectural level, Boolean masks are typically applied to all intermediate values. This causes a relatively high overhead (especially in non-linear functions like the AES S-box), which is not acceptable for a passive RFID tag. Also masking countermeasures on the cell level require high additional costs. For example, the logic style Masked Dual-Rail Precharge Logic (MDPL) proposed by [PM05] that has been invented to protect circuits at the cell level requires 4.5 times more chip area as standard Complementary Metal

Oxide Semiconductor (CMOS) logic.

In contrast to masking, hiding countermeasures try to break the link between the processed data and the power consumption. The goal is to consume either random power in each clock cycle or to consume equal power in all clock cycles. In the amplitude dimension this is achieved by increasing the noise or by reducing the data-dependent signal. However, increasing the noise also means to increase the power consumption, which is very limited in passive RFID tags. Reducing the data-dependent signal is typically achieved by using a Dual-Rail Precharge (DRP) logic style like Sense Amplifier Based Logic (SABL) proposed by [TAV02]. As already explained, secure logic styles require a significantly higher chip area and have a high power consumption.

The most efficient hiding countermeasures for RFID tags work in the time dimension. In these methods, the attacked operations of the cryptographic algorithm are executed at different moments in time during each execution. This can be implemented by shuffling or by randomly inserting dummy operations. The basic idea of shuffling is to randomly change the sequence of the operations such that in each execution the attacked intermediate result occurs at a different time location. Unfortunately, this measure only works to a certain extend because the different possibilities are limited depending on the executed algorithm. In case of AES this works on byte level where 16 different positions are feasible.

An alternative and an extension of the randomization degree is the insertion of dummy cycles. In this method, so-called dummy operations are inserted before, during and after the execution of the actual operation. It is important that the total number of dummy cycles and also the cycles for the operation itself are constant to not allow an attacker to get any information about how many dummy cycles have been introduced or which data is processed. Before the execution, a random source decides at which positions the additional operations are inserted.

The advantages of countermeasures that work in the time domain are that the additional hardware resources are very low and the power consumption is nearly not increased. Only the controlling effort of the cryptographic module is higher. The additional runtime of the algorithm is for RFID tags, which have very low data rates, typically no problem. This makes randomization in time to the best solution for low-cost countermeasures in passive RFID tags. An interesting solution has been presented in [FP08]. This paper shows the application of several low-cost countermeasures that have been optimized for the use in hardware modules for passive RFID tags.

2.6 Summary and Conclusions

In this chapter, we introduced side-channel attacks that have been performed on RFID devices. First, we measured the power consumption of RFID tags at the antenna connections and by probing internal chip signals. Second, we performed several electromagnetic attacks by using special setups to increase the SNR of the measurements. In particular, we applied the Helmholtz arrangement,

separated the chip from the antenna, and performed frequency-selective filtering using a receiver. Third, we identified a data-dependent timing behavior of RFID tags that can be exploited. All performed attacks have been successful which emphasizes the need of appropriate countermeasures against these attacks.

This chapter identified the potential threat of side-channel attacks on RFID devices. We have shown that the information leakage can not only be exploited by evaluating side-channel signals of internal chip components but also from leakages in the antenna connections. Furthermore, using the Helmholtz arrangement, we have demonstrated that the magnetic-coupled reader field contains data-dependent signals of RFID tags which can be exploited even from a distance.

Best results have been obtained using electromagnetic attacks. If tiny magnetic near-field probes are used, the impact of the reader field can be significantly reduced. The pre-processing of measured side-channel traces is mandatory for all attacks to be successful. Our experiments showed that calculating the envelope signal of acquired traces effectively helps in reducing the noise due to misalignments. This, however, can be only applied for devices which provide data-dependent leakages in a low frequency band.

In Table 2.1, an overview of the presented side-channel analysis attacks is given. The table lists the advantages and disadvantages of each method to characterize their properties and performance efficiency.

Table 2.1: Efficiency of the described SCA methods performed on RFID devices.

	Advantages	**Disadvantages**
Power	◦ higher reproducibility due to a contact-based measurement setup	◦ insertion of a measurement resistor ◦ direct contact to the chip necessary (destructive) ◦ no selective measurement of chip components
EM	◦ (remote) attacks at a distance ◦ hard to detect (undetectable) ◦ non destructive ◦ selective measurement of chip components	◦ weak signals in remote attacks (amplifiers needed) ◦ noise of the proximity ◦ accurate positioning of probes required
Time	◦ RFID devices often process data serially	◦ higher sampling rates required

3

Fault-Analysis Attacks on RFID Devices

The greatest of faults, I should say, is to be conscious of none.
— Thomas Carlyle

The main intention of this chapter is to investigate the susceptibility of RFID devices against fault attacks. This allows the verification whether the threat of fault attacks on such kind of devices is realistic or not.

First, we will introduce fault attacks in general. We will describe the main principle of such attacks and give related work on that topic. After that, we will describe different fault-injection setups to perform such attacks on RFID devices. In our experiments, several tags from various vendors have been examined including HF and UHF tags. The tags include neither cryptographic primitives nor countermeasures against fault-analysis attacks. Instead, we targeted the writing of data since this operation is considered critical with respect to power consumption and execution time. We performed attacks using spike and glitch injections, electromagnetic interferences, optical inductions, and antenna tearing. For spike and glitch attacks, we varied the power supply and clock source of the tag during its computations. Electromagnetic interferences, in contrast, were conducted by using a high-voltage generator. Faults by optical inductions have been obtained by irradiating the chip with a simple laser diode. Finally, we caused faults by temporarily interconnecting the antenna pins of the RFID chip.

This chapter is based on results of a joint work together with Jörn-Marc Schmidt, Thomas Plos, and Martin Feldhofer. Parts of this work have been published at the CHES conference in 2008 [HSP08], the ECCTD conference in 2009 [HSP09], and in the International Journal of Security and Networks in

2010 [HPF10].

In the following, we will present the results of our investigations. At first, we will show that all investigated RFID tags are vulnerable to the performed attacks. In particular, we show that our fault-injection methods allow the prevention of writing data into the tag memory. Even worse, they allow the writing of faulty values. In both scenarios, the tags confirm the write operation to be successful. Our findings led us to the conclusion that countermeasures against such attacks have to be included, especially if they are applied in security-related applications.

3.1 An Introduction to Fault Attacks

In contrast to side-channel analysis attacks, which exploit physical properties by means of passive measurements, fault analysis attacks extract secret information through active interferences. By maliciously manipulating the working conditions of a cryptographic device during its operation, erroneous behavior can be provoked. This erroneous behavior is then exploited in an attack to reveal secret information. In general, fault attacks can be divided into *non-invasive attacks*, *semi-invasive attacks*, and *invasive attacks*.

Non-invasive attacks. Non-invasive attacks leave no damage of the examined device. Such attacks physically stress the device by applying techniques like electromagnetic interferences. R. Anderson and M. Kuhn [AK96, AK97] have been one of the first who injected non-invasive glitches in the clock signal and induced spikes in the power-supply line of commercially-available smart cards. J. J. Quisquater and D. Samyde [QS02] have applied non-invasive fault attacks by injecting electromagnetic spikes into cryptographic devices. They use eddy currents to influence the behavior of the circuit.

Semi-invasive attacks. Semi-invasive attacks require the decapsulation of the chip to have a direct sight to the surface of the chip die. This modification allows to induce faults by exposing the chip to light (optical beams). S. Skorobogatov and R. Anderson [SA03] have demonstrated that the light of a laser pointer can be utilized to perform successful fault attacks on a cryptographic device after decapsulating the chip.

Invasive attacks. Invasive attacks are by far the most powerful fault attacks. For these attacks, the chip is decapsulated and its surface is electrically contacted with special probes (a probing station is often used in practice to establish a connection to internal wires of a circuit). In that way, secrets stored on the device can be directly read out and also modified by injections. In particular, the use of a Focused Ion Beam (FIB) allows to modify the chip layout by cutting and reconnecting specific wires of the circuit. Unlike non-invasive and semi-invasive attacks that can be conducted at moderate costs, invasive attacks require expensive equipment that is only available at special laboratories.

Fault-injection methods can be further separated into their impact and effect on the device. They can be induced *globally* or *locally*. Global fault-injection methods influence the entire device and are therefore quite imprecise. Local fault-injection methods, in contrast, affect only specific parts of the device. There, it is possible to focus on certain regions that are assumed to contain sensitive information.

The resulting faults can be classified into three types: *transient*, *permanent*, and *destructive*. Transient faults are faults that affect the device only once. That means that after an injection, the device can compute the same operation again without any faulty behavior. This is one of the most powerful type of faults because it does not leave any damage or any marks on the circuit. Examples of transient faults are faults that cause bits of registers to flip or that over-jump individual instructions. Permanent faults, in contrast, remain after an injection until the device performs a reset. An attack can exploit such permanent faults by collecting many faulty computation results through several execution runs of an algorithm. A typical scenario of permanent faults are the modification of volatile memory entries which are only initialized after a chip reset. Destructive faults remain after injection even after a reset of the device. These types of faults manipulate the device permanently and cannot be reversed anymore. Examples are when individual bits of a device get *stuck at* a certain value (0 or 1) after a fault injection.

In the following, we describe various fault-injection methods that have been used in our experiments. The setups are composed of several components: a PC, an RFID reader, a trigger device, and the device under attack. The PC controls the devices and the overall fault-injection process. For this, we used Matlab [The] and implemented Matlab scripts to establish a connection to the reader over the serial interface as well as to provide useful functionalities like plotting and post-processing facilities for the attack. For almost every attack, a trigger device has been used that is capable of generating a trigger signal after a reader-to-tag communication. It is used to provide a trigger event that starts the fault-injection process. After a specific offset time, a fault is injected into the device. We increased the offset time in steps of about 300 ns and injected several faults during the response time of an RFID tag. In fact, a tag performs computational work during reader request and tag response. Figure 3.1 shows the location of the response time.

The following fault-analysis attacks are non-invasive[1] and do not leave any damage of the chip. Furthermore, all faults are transient and do not make any permanent remainders. Thus, a fault can be injected not only once but several times.

3.2 Spike and Glitch Attacks

Among the most popular fault attacks against cryptographic devices are spike attacks and glitch attacks. The main difference of these two kinds is which

[1] Our attacks do not modify the chip layout but only the RFID-tag inlay.

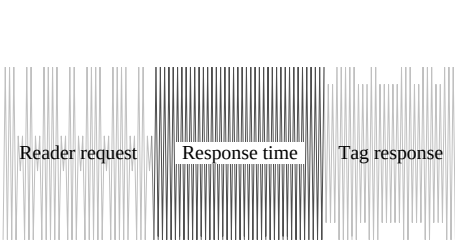


Figure 3.1: The tag performs computational work in the response time that is located between reader request and tag response.

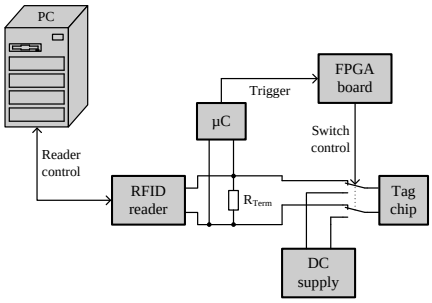


Figure 3.2: Setup for injecting over-voltage spikes in RFID tags.

external supply line of a device is interfered by the attack. Spike attacks basically affect the power-supply line of a device while glitch attacks interfere the clock-signal line.

Essentially, integrated circuits are designed to operate within a certain supply-voltage range. For CMOS devices, this range typically covers 1.2 V to 5 V. If the supply voltage exceeds the power-supply deviation tolerance of the device, the device might produce faults during the execution of algorithms. So-called induced over-voltage (or under-voltage) spikes can cause the chip to perform erroneous computations and can modify the execution path of the device. Glitch attacks, in contrast, modify the clock signal that is fed into the device. This can be a higher frequency signal which can cause a timing violation because of critical path or due to setup and hold timings of the hardware components. Due to the increased frequency, (combinatorial) computations or signals become not stable before the next active edge of the clock. Thus, intermediate values get wrong and the result is faulty.

One of the first publications that demonstrated the effectiveness of spike and glitch attacks, has been published by R. Anderson and M. Kuhn [AK97] in 1997. They injected spikes and glitches in the power-supply line and clock source of a smart card and performed successful attacks on the Data Encryption Standard (DES) [Nat99] and the RSA cryptosystem [RSA78]. Such algorithms are typically used in practice and are applied in a wide range of security applications. D. Boneh et al. [BDL97] as well as E. Biham and A. Shamir [BS97] presented theoretical fault attacks on DES and RSA implementations in the same year. E. Biham and A. Shamir first introduced the name Differential Fault Analysis (DFA) and showed how to extract the secret key out of symmetric primitives using fault attacks. Practical experiments on smart cards have been described by O. Kömmerling and M. Kuhn [KK99] in 1999. They showed various techniques including glitch attacks in order to extract protected data from commer-

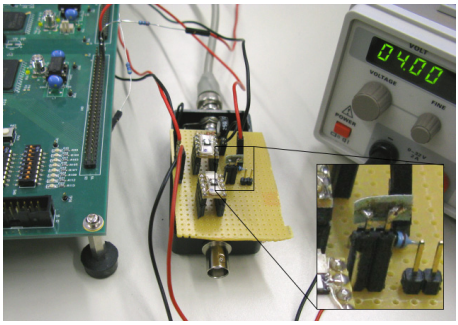


Figure 3.3: Setup of the fault attack to induce an over-voltage spike into the antenna connections of an RFID chip.

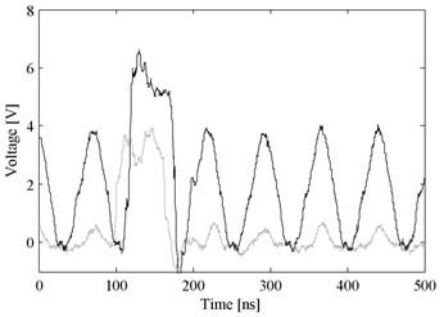


Figure 3.4: Injection of an over-voltage spike during the writing of data into the tag memory. The injected spike is drawn in black, the trigger signal is drawn in gray.

cial smart-card processors. Glitches produce malfunctions and cause chips to overjump individual instructions. This behavior can be exploited especially in attacks which try to protect an algorithm by conditional branches. The injection of faults can prevent the execution of such branches.

3.2.1 Description of Our Fault-Injection Setup

In order to perform spike and glitch attacks on RFID devices, an appropriate fault-injection setup is needed. For this, we used a setup that is similar to the power-consumption setup described in Chapter 2. The setup is composed of several components: a PC, an RFID reader, an impedance-matching resistor, an RFID tag, a microcontroller with an analog front-end, an FPGA board, a DC power supply, and two high-speed multiplexers. A schematic view of the setup is shown in Figure 3.2.

As already described before, we used Matlab [The] in order to control the overall fault-injection process. A Matlab script has been written that controls an RFID reader over a serial connection. Next to the serial interface, the reader provides a 50 Ohm output for an external antenna. Instead of an antenna, we connected a 50 Ohm impedance-matching resistor that is used to match the impedance of the reader-antenna output. After that, we connected the RFID tag in parallel to that matching resistor using a two wire cable. For this, the tag has been separated from its antenna and the chip has been soldered on a two-pin strip. The antenna output of the reader is further connected to a trigger device that controls the fault-injection time. The trigger device mainly consists of a fast envelope detector (analog front-end) and is used to transform the analog signal of the reader into digital signals. The digital trigger signal is then fed into an 8-bit microcontroller (AVR ATmega128 from Atmel [Atm07]) which has been programmed to set a dedicated output pin to high when a write command

Figure 3.5: High-voltage generator that produces up to 18 kV output.

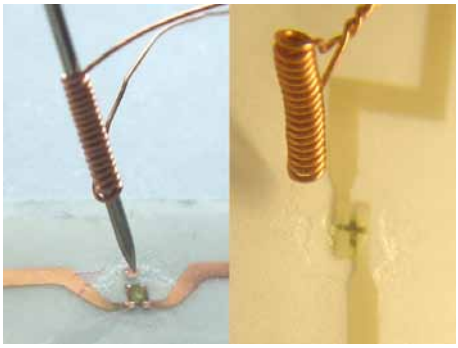


Figure 3.6: Electromagnetic fault-injections using tiny probe coils.

is received. Next to the microcontroller, an FPGA board is used that is able to generate signals of very short duration. In addition, it is used to provide different trigger delays and trigger durations in order to vary the fault-injection time and period for a successful attack. We have used the Side-Channel Attack Standard Evaluation Board (SASEBO) [Sid] for this that features two Xilinx VirtexTM-II Pro devices. One FPGA device has been programmed to concurrently control two ISL54200 high-speed multiplexers. The multiplexers are used to connect the tag to either the reader or the DC power supply. If the multiplexers are activated, the tag is no longer powered by the reader and an over-voltage spike is injected into the antenna-pad connections that causes the tag to perform faulty operations. A picture of the RFID chip and the surrounding components such as the FPGA board and the DC power supply is depicted in Figure 3.3.

Figure 3.4 shows the injection of an over-voltage spike during the writing of data into the tag memory. The injected spike is drawn in black, the trigger signal is drawn in gray.

3.3 Electromagnetic Interferences

A fast-changing electromagnetic field induces current into conductors. Such a field is generated by current that is flowing through a coil. The windings of the coil, the size, and the distance from the coil to the chip surface define the pulse strength and efficiency of the electromagnetic fault injection.

Electromagnetic attacks are non-invasive attacks and allow fault injections to be performed without direct contact to the device. No chip-decapsulation procedure has to be applied in practice. However, the position of the probe often requires the accuracy of a probing station. Depending on the position and distance of the probe, *globally* as well as *locally* injected faults can be achieved.

J. J. Quisquater and D. Samyde [QS02] have been one of the first who published results of electromagnetic fault attacks on smart cards. They used a

probe coil with several hundred windings and induced eddy currents inside the chip. They have been able to insert transient as well as permanent faults in the memory of the chip.

3.3.1 High-Voltage Generator

A high-voltage generator has been built to inject faults during the writing of data into the tag memory. The device is shown in Figure 3.5. It is capable of generating electromagnetic sparks of high intensity. The circuit consists of a digital part that is used to produce a pulsating square wave of about 100 V. This pulse is then amplified using a DC voltage converter and a charge-pump circuit. Using this setup, the injection voltage can be raised up to 18 kV. The injection is controlled by a high-voltage relay. The output of the generator is then connected to a probe coil. As soon as the current flows through the coil, a current is induced into the RFID chip which influences the processing of operations. In Figure 3.6, two probe coils are shown that have been placed on top of RFID chips.

3.4 Optical Fault-Injections

Light that hits special regions of a chip induces a current[2]. This current is often referred to as Optical Beam Induced Current (OBIC). Internal transistors get switched by the light and cause faults during the processing of data [TTO97]. In general, optical faults are semi-invasive and need the decapsulation of the chip. A light beam has to be focused directly on the surface of a chip die. Thus, it is essential to have intervisibility to regions that are intended to be attacked. In view of RFID tags, many chips are only covered by a transparent Polyethylene Terephthalate (PET) inlay. Parts of the chip are also hidden by the antenna circuit. Remaining PET layers, adhesive, and dirt can be either removed by carefully scratching them off or by using chemicals to provide a direct line of sight to the chip die. For tags that use transparent inlays, optical inductions are performed innately without further depackaging. In this context, they are therefore considered to be non-invasive.

One major advantage of optical inductions is the fact that the fault can be accurately adjusted by the fault-injection location, the focus, the size, and the strength of the used light beam. Due to the precise configuration, it poses one of the most powerful attacks in practice.

S. Skorobogatov [SA03] has first demonstrated the potential of optical fault-injections on secure microcontrollers and smart cards. They showed that it is possible to cause simple transistors to conduct by illuminating the chip with a focused laser beam. J. Blömer and J.-P. Seifert [BS03] presented optical as well as electromagnetic fault attacks on AES. The attack allows the extraction of the complete 128-bit secret key of a sealed tamper-proof smart card.

[2]Depending on the wavelength of the light, a line-of-sight to the NP junction of the target transistor is required to cause an electron reaction.

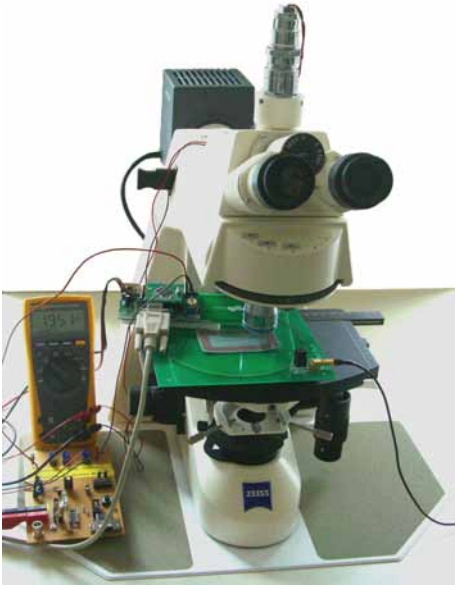


Figure 3.7: Optical fault injections on an RFID chip using a focused laser beam.

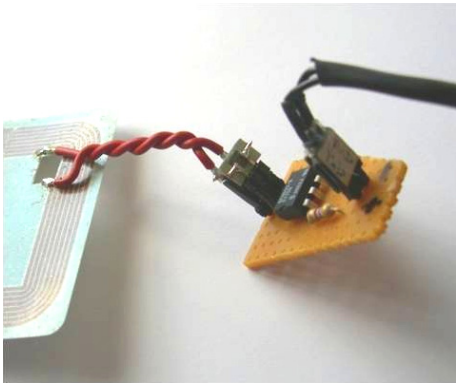


Figure 3.8: Setup of a tearing attack on an HF RFID tag.

3.4.1 Laser-Beam Injection using a Microscope

In order to inject faults into RFID devices, we have used a microscope to focus a light beam to specific regions on the chip. The microscope has an integrated camera port where a camera can be connected to take pictures of enlarged chip regions. Instead of the camera, we connected a laser diode which provides an optical output power of about 100 mW. The emitted light has a wavelength of 785 nm. Furthermore, a collimator lens is used to parallelize the laser beam. The focus has been realized through an optical objective which offers a magnification of 50 diameters. In Figure 3.7, the setup for the optical fault-injection attack is given. The focused laser beam illuminates an RFID tag that lies upon an RFID-reader antenna. The communication of the reader is eavesdropped by a trigger device. The trigger device listens to reader requests and starts the illumination of the chip by turning on the laser diode. For the experiments, we have varied the illumination starting time as well as the illumination duration and used different RFID devices to evaluate their susceptibility.

3.5 Tearing Attacks

Another possibility to cause faults on RFID devices is to tear off the antenna during the execution of sensitive operations. So-called tearing attacks are usually known from contactless smart-card systems, where smart-card implemen-

tations have to take care of sudden power losses during sensitive computations. E. Hubbers et al. [HMP06] presented such attacks on different Java card platforms. They made several experiments on cards from various vendors and focused on the transaction mechanism. By applying tearing attacks, they showed that the cards perform faulty computations and return unpredictable results which might can be exploited by an attacker.

In view of RFID, tearing attacks can be similarly performed by temporarily conducting the antenna pads of the tag. This bypasses the antenna connection for a certain amount of time which may cause the chip to perform erroneous computation results. In contrast to spike attacks, where an overvoltage spike is used to cause faulty computations, the power supply is cut off for a certain amount of time so that not enough power is available to finish the operation accordingly.

The attacks have been performed as follows. First, we have separated the chip from the antenna. Between the chip and the antenna, we have placed an optocoupler that is used to temporarily interconnect the antenna pins of the RFID chip. Optocouplers use a short optical transmission path that allows the transmission of signals without having electric contact. This optocoupler is controlled by the same trigger device that was also used for the other fault-attack experiments described before. In Figure 3.8, the setup of the tearing attack on an HF RFID tag is shown.

As a second step, a write command is sent by the reader that causes a trigger event during the writing of data into the memory. This trigger event switches the optocoupler during the response time which interconnects the antenna pins of the chip. Accordingly, the chip suddenly looses the power supply and is not able to finish the computation without a faulty result.

3.6 Results

In our experiments, different kinds of passive tags have been analyzed. In addition, various faults have occurred that are described in the following.

Generally, five fault types have occurred and are listed in Table 3.1. Each tag has its own behavior pattern such as the writing time, the duration of writing, and the writing strategy (e.g. erasing the memory before writing). There are tags that are more sensitive to certain classes of faults while there exist others that are less sensitive. Once the offset and the length of the fault-injection trigger is adjusted accordingly, all examined tags show the same faulty behavior and the same results have been obtained during all our tests. Note that the investigated tags are from different vendors but are compliant to the same RFID-protocol standard. Two types of faults occurred that are also defined in the standard. We have denoted these faults by *Unconfirmed Lazy Write* and by *Unconfirmed Successful Write*. *Unconfirmed Lazy Write* indicates an unsuccessful write operation where the tag does not confirm the write-operation request. The value of the tag memory remains untouched. In contrast, *Unconfirmed Successful Write* represents a successful write operation but the tag does not confirm the opera-

tion. Though, the new value is stored in the memory. In case of errors, protocol standards provide a certain waiting time in which tags can send an error response like *insufficient power*.

Table 3.1: Fault types and resulting EEPROM values.

Fault type	EEPROM value
Unconfirmed Lazy Write	old
Unconfirmed Successful Write	new
Unconfirmed Faulty Write	influenced by adversary
Confirmed Lazy Write	old
Confirmed Faulty Write	influenced by adversary

Unconfirmed Faulty Write, in contrast, indicates the behavior where the tag does not confirm the reader request but different values are stored in the memory. These values are not random and depend on the trigger delay, the trigger duration, the original memory value, the value that has to be written, and the type of fault injection. Another interesting fault that has occurred has been denoted by *Confirmed Lazy Write*. Thereby, the tag did not perform the memory writing but confirms the operation to be successful. At last, we have observed the case where the tag writes different values to the memory but confirms the operation to be successful. This case is denoted by *Confirmed Faulty Write* and is one of the most critical type of faults.

3.6.1 Global Fault Injections

For all injection methods that induce a fault globally, the same results have been obtained. Global fault-injections have been obtained by any method described before. In the following, we will first discuss the impact of the duration of a fault injection as well as the injection time (offset of the trigger signal). After that, we will give results of the performed global fault-injection attacks.

The duration of an injected fault is an important factor for an attack. If it is chosen too short, it does not have an impact on the device under attack. If the fault duration is chosen too long, the tag performs a reset due to the absence of power supply and will not answer to the reader request. The time when the tag actually causes a reset due to these induced fault depends on several factors. These factors are, for example, the field strength of the reader device and the distance between the tag and the reader, respectively, or the fault-injection technique (spike, glitch, EM induction, optical inductions, or antenna tearing). If the distance between the tag and the reader is chosen short, the duration of the fault has to be longer as compared to the scenario where the distance between the tag and the reader is chosen long. In general, the more power is available for the tag, the longer must be the fault-injection duration to cause the chip to fail. However, while the duration of the fault is important for spike and glitch injections, optical inductions, and antenna tearing, it is not

for electromagnetic injections. The high-voltage generator produces EM pulses which have a fixed pulse width of only a few nanoseconds. Our experiments have shown that even one pulse is sufficient to force a reset of the tag. For antenna tearing, the duration of the fault injection constitutes the time in which the tag is not supplied by the field anymore. For optical inductions, the duration of the fault injection defines the period of time when the laser diode is illuminating the chip. We finally have chosen a fault-injection time of about 100 µs.

Next, we have varied the trigger offset by starting at the beginning of the response time. Figure 3.9 shows different types of occurred faults. Depending on the offset value, we observed the occurrence of three different fault types: *Unconfirmed Lazy Write*, *Unconfirmed Faulty Write*, and *Confirmed Successful Write*. In fact, if the offset is chosen small, the reset is performed before the writing of data. Thus, the tag does not send an answer anymore and the content of the memory keeps unchanged. If the offset is chosen very high, the tag performs a reset after the writing of data. The tag is able to write the new memory content but is disturbed before sending the answer. However, if the offset of the fault trigger is chosen to occur during the writing of data, the content of the memory becomes modified. In addition, varying the offset very slightly leads to different memory contents. In Figure 3.10, the memory values are depicted as a function of the trigger offset. The black curve has been achieved by initializing two bytes of the memory with zero and setting all bits of them to one during an *Unconfirmed Faulty Write* operation. The gray curve describes the same for writing zeros to the memory that was first initialized with ones. It can be observed that the data bits are serially written into the memory and that different bits are flipped at different positions in time. The more time is proceeded the more bits are actually written. In fact, 16 bits are written while the Most-Significant Bit (MSB) makes the highest value step. The Least-Significant Bits (LSB) have only a small impact on the written value and thus cause only small value steps which are not clearly discernable in the given figure. Nevertheless, note that we have influenced the bits not sequentially (i.e. from the LSB to the MSB) but we have rather influenced specific bits at specific points in time.

With the help of an automatic sweep, we are able to detect the writing of data into the memory within a few minutes. It is possible to determine the time when the writing of data starts and how long it takes. It is further possible to detect if the memory content is cleared before the real writing of data. This allows fingerprinting of tags by identifying device-specific patterns for operations like writing to the memory.

While the same results have been obtained for all injection methods, optical inductions have led us to further interesting findings. Besides *Unconfirmed Faulty Write* faults, we have been able to produce *Confirmed Lazy Write* and even *Confirmed Faulty Write* faults. This has been achieved by accurately adjusting the duration of the fault injection to a limit where the tag has barely enough power to confirm the write operation but it is not able to write the exact value. The tag either keeps the old value or it stores a different one. However, by choosing the right time and by varying the trigger offset we have been able

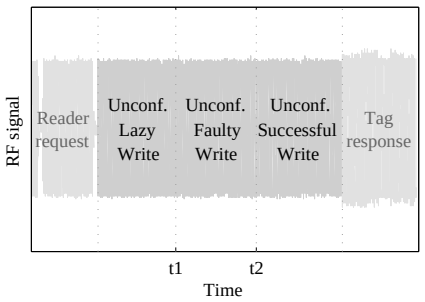


Figure 3.9: Types of faults occurred at different points in time within the response time.

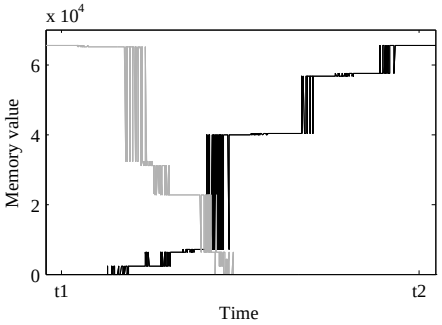


Figure 3.10: Memory value during *Unconfirmed Faulty Write* after writing two different values (black curve: $0x$FFFF; gray curve: $0x$0000) by varying the delay of the fault injection.

to roughly influence the modification of individual bits that have to be written. Following these facts, it appears that this allows the modification of memory content at several points in time during the writing of data. Hence, it is possible to bypass common security features which make use of backup facilities or memory-shading techniques. It might be further possible to skip the increasing or decreasing of counter values. Counters are commonly used in cashing applications or they are used to limit the number of authentication steps to avoid differential side-channel attacks.

As a simple countermeasure against these attacks, a comparison of the value that has to be written and the actually written value becomes reasonable. However, if the register that stores the new value is modified by faults before the writing into non-volatile memory, a comparison does not help to detect the failure. This article focuses only on the modification of writing into the non-volatile memory and does not analyze the susceptibility of faults on register values. This point keeps unclear at this stage but is marked for future work.

3.6.2 Local Fault Injections

Next, we focus on local fault injections using an optical laser beam. In fact, the injection of local faults offers more control for an adversary. Depending on the position of the light beam, different components of the integrated circuit are affected. The occurred faults range from simple resets to definite modifications of tag memory contents. We have been able to generate all kinds of faults that have been described in the section above.

By increasing the power of the light or by broadening the beam of the laser, simple resets have been enforced. As soon as the laser beam has been focused

on the memory control logic, various faults are initiated such as they have been obtained by global fault injections. All in all, for local fault injections the timing of faults is not that relevant as the issue of fault-injection location. Once the laser beam has been focused to a specific position, injecting faults is rather easy. The fault type *Confirmed Faulty Write* has been achieved without accurately adjusting the illumination time to a certain period. However, this convenience is compensated by higher costs for the equipment.

3.7 Countermeasures against Fault-Analysis Attacks

Countermeasures for RFID devices against fault attacks are very similar to those which have been integrated on (contactless) smart cards. Basically, there exist three different concepts for such countermeasures. The first concept is to prevent the injection of faults. This can be realized for example through passive or active shielding. A mesh of power lines is placed on top of the chip surface. This countermeasure makes electromagnetic attacks much more difficult to perform. Another way to prevent the injection is to integrate different sensors that check the power supply and the clock signal for critical modifications. An alarm might be triggered in case of an injection that causes the chip to jump into a specific safe state. There also exist detectors for optical light and temperature variations that look for deviant operating conditions. In security-related applications, some devices also include a self-destruction capability in case of an external intrusion detection. In view of passive RFID tags, these kind of countermeasures are quite effective but need a high amount of overhead in respect of power consumption and chip size. They are therefore often omitted in practical implementations.

The second concept is to distribute computations of sensitive data over a certain period of time. In fact, this does not prevent the fault injection itself but reduces the exploitation of sensitive information. For RFID devices, this countermeasure offers a potential and effective way to prevent the extent of information extraction by requiring an inessential amount of implementation costs.

The third concept makes use of redundant computations. This is indeed one of the most intuitive and usable ways in practice. Intermediate values and results are computed twice and checked at the end of both computations. The computation can be calculated consecutively on the same hardware unit or it can be calculated in parallel. The implementation of the parallel version obviously needs twice the area which often discourages the use of concurrent executions. Especially for RFID implementations, the consecutive computation is attractive due to two reasons. First, time is typically not a critical factor in RFID applications. The recalculation of results is therefore possible without any functional constraints. Second, the additional overhead in terms of chip area is only marginal compared to its fault-protection capability. Alternatively, error-detecting codes can be applied. The codes calculate the signature of sensitive

data and they concurrently check the values by using for instance parity bits.

While there exist many proposals and solutions for fault protection in practice, there will be no effective and satisfying protection against powerful adversaries. The history already illustrated the difficulty of preventing attacks and intrusion techniques also on security-related devices. Countermeasures may not provide complete and perfect protection for a given system but they can make attacks considerably harder to perform in practice.

3.8 Summary and Conclusions

In this chapter, we made fundamental observations about the vulnerability of commercially available RFID tags. We conducted several fault attacks by applying different setups for spike and glitch injections, EM interferences, optical inductions, and antenna tearing. Since all examined tags do not provide any cryptographic capabilities, we targeted the writing of data into the tag memory. Our results show that the attacks allow the prevention of writing data into the tag memory and, even worse, to allow the writing of faulty values.

Best results have been obtained by optical inductions. By illuminating tags with a focused laser beam, we caused the tags to write faulty values into the memory by confirming the write operation to be successful. The attacks allow therefore to modify the memory content of tags during the write operation without being detected.

In Table 3.2, an overview of the presented fault attacks is given. The table lists the advantages and disadvantages of each method to characterize their properties and efficiency.

We conclude that many RFID devices nowadays provide potential weaknesses in terms of security. As a future work, cryptographic-enabled RFID devices have to be analyzed due to faults to validate their susceptibility against these attacks. Countermeasures have to be considered in implementations especially in applications where cryptographic services are used to provide a certain level of security.

Table 3.2: Efficiency of the described fault-injection methods performed on passive RFID devices.

	Advantages	**Disadvantages**
Spikes/Glitches	◦ easy modification of data ◦ simple setup (unexpensive)	◦ direct contact to the chip (destructive)
EM	◦ non invasive ◦ allow (remote) attacks at a distance	◦ high voltages (dangerous) ◦ appropriate equipment required ◦ can lead to a destruction of the chip
Optical	◦ selective location of fault injection ◦ allow attacks at low distance	◦ decapsulation maybe necessary ◦ more expensive equipment needed
Antenna Tearing	◦ low cost	◦ removing chip from antenna (destructive)

4

Attacks on Public-Key Enabled RFID Devices

Most security failures in its area of interest are due to failures in implementation, not failure in algorithms or protocols.
— NSA

In this chapter, we investigate practical side-channel attacks on public-key enabled RFID devices. We performed power-analysis attacks and electromagnetic-analysis attacks on an RFID-tag prototype that implements the Elliptic Curve Digital Signature Algorithm (ECDSA) in hardware. As opposed to existing side-channel attacks on Elliptic Curve Cryptography (ECC), we present an attack that targets the private-key multiplication in ECDSA instead of targeting the ephemeral key during scalar multiplication. Our results show that the private key of ECDSA can be successfully revealed with less than 2 000 power traces or 10 000 acquired electromagnetic traces. Revealing the private key of ECDSA allows the forging of digital signatures and allows adversaries to illegitimately authenticate themselves to verifiers or even proof the origin of RFID-labeled goods. The integration of countermeasures against these attacks is therefore a mandatory part in the design of a secure RFID system.

The following chapter contains results of a joint work together with Marcel Medwed, Daniel Hein, Johannes Wolkerstorfer, and Christoph Nagl. Parts of the outcomes have been published at the Applied Cryptography and Network Security (ACNS 2009) conference [HMHW09], the International Conference on Security and Cryptography (SECRYPT 2009) [Hut09], and the Workshop on RFID Security (RFIDSec 2009) [NH09].

In the following, we first introduce to public-key cryptography. We define and describe a general model for state-of-the-art public-key techniques which particularly provide authentication as a cryptographic service. After that, we present our attack and give an overview about related work on side-channel attacks on ECC implementations. Next, we give details about the used RFID prototyping platform. Furthermore, we will describe the measurement setup used in our experiments. We show how to improve practical attacks on RFID devices by applying a trace-decimation technique that increases the signal-to-noise ratio of side-channel measurements. Finally, the result of our attack is given and discussed. A summary of the work is given in the conclusions.

4.1 Public-Key Cryptography

There exist many applications nowadays that have to provide security services such as *authentication*, *confidentiality*, *non-repudiation*, and *data integrity*. Although these services can be provided by using symmetric cryptography, public-key techniques have often been preferred and used in practice to tackle the problem of key distribution. This is especially important in view of RFID systems where tags are deployed in a large scale. There, tags can be shipped along with a secretly kept private key whereas readers can use the corresponding public key to verify the authenticity. In such open-loop systems, it seems almost inevitable to integrate public-key techniques to provide these services.

Cryptographic services can not only be achieved by a specific public-key technique. There exist various public-key techniques which lead to the same cryptographic service. Authentication, for example, can be achieved by various cryptographic protocols that are based on different schemes and primitives. Both digital-signature schemes and zero-knowledge proof-of-knowledge schemes can be used in protocols to achieve the same security goal of entity authentication, for instance. Although many of these protocols have been standardized in the last decade, less effort has been made to uniformly classify these protocols according to their underlying public-key technique. Hence, we first describe a general model to characterize different state-of-the-art public-key techniques. The model separates all techniques into four levels: *services*, *protocols*, *schemes*, and *primitives*. Figure 4.1 shows the natural structure of this model as an instance of authentication.

4.1.1 Services

Authentication has become one of the most important services of cryptographic applications. It is the process where someone or something is confirmed to be authentic and that a verifier is assured of the identity of a prover. The authentication service can be separated into two main types: *entity authentication* and *message authentication*. Entity authentication is referred to as a real-time service where both parties the prover and the verifier actively participate in a communication. Thus, the identity of the prover is guaranteed in a timeliness

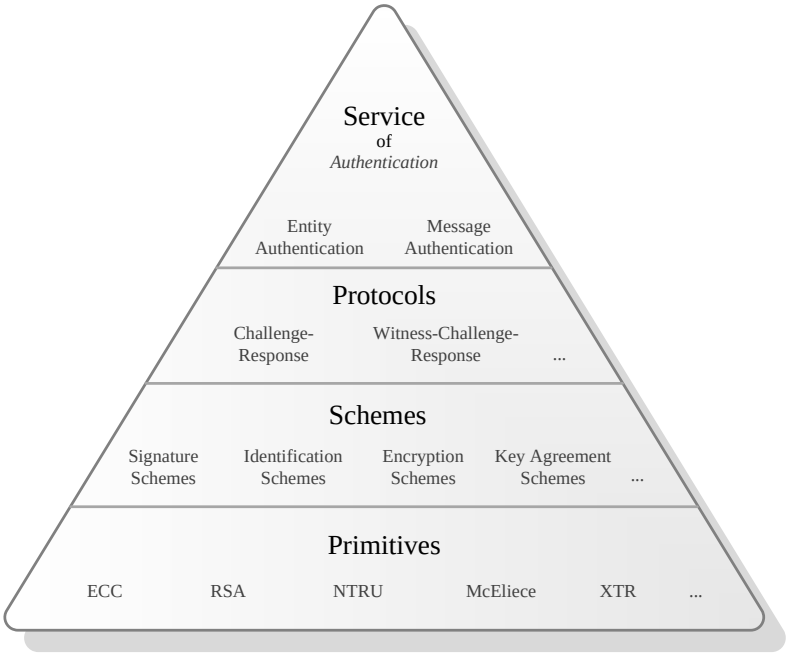


Figure 4.1: The cryptographic service of authentication can be obtained by different public-key techniques.

fashion. In contrast, message authentication refers to the service where a verifier gets assurance of a dedicated message that was generated by the prover. The difference between entity and message authentication is therefore as follows. In the first case, an entity or person is authenticated exactly during an actual communication process. The proof of authentication is not transferable this scenario. In the second case, a prover digitally signs a message which can be verified even after the actual communication process. This is useful if one party is not active in the communication, which may happen in Internet applications. However, the latter case implicitly provides two additional cryptographic services that are data integrity and non-repudiation of the signed message which makes the use of digital signatures applicable in a broader range of applications.

4.1.2 Protocols

Protocols are typically used to provide one or more cryptographic services. They define a sequence of steps for two or more entities that would like to achieve these services. Such protocols differ in several properties. First, they can be built upon different schemes. Entity authentication protocols, for example, can be based upon encryption schemes, signature schemes, or identification schemes. Protocols that provide message authentication can be constructed by signature schemes or schemes that apply message-authentication codes (MACs), for ex-

ample. Second, protocols may differ in the number of message passes between the involved entities. The number of message passes of a protocol actually depends on the required message passes of the underlying schemes and also on the protocol-specific passes that are needed to achieve a certain service. Challenge-response protocols usually need two message passes but there exist also other protocols that need more or even less message passes than the sum of message passes of the individual schemes they involve. Third, there are protocols that provide services for all or a subset of involved parties. In an unilateral authentication protocol, only one entity is authenticated, whereas in a mutual authentication process both parties get assured to be authentic [MvOV97].

4.1.3 Schemes

Schemes are the basic building blocks of cryptographic protocols. They provide a set of cryptographic operations and methods that are typically combined within a protocol in order to achieve a certain security service. One of the most commonly used schemes are encryption schemes, signature schemes, identification schemes, and key-agreement schemes. Encryption schemes provide encryption and decryption routines and make use of encryption primitives such as ElGamal or RSA. Signature schemes provide a signature generation and signature verification operation. These schemes use additional cryptographic functionalities such as hash or redundancy functions. Identification schemes, in contrast, offer a prove and verify operation. They typically provide additional functionalities such as random numbers that are needed to compute the challenges. Key-agreement schemes offer methods to agree on a common session key. In addition to the described operations, schemes in general also offer methods for key management such as the generation and verification of public keys.

4.1.4 Primitives

The lowest level of our model are cryptographic primitives. Primitives are cryptographic algorithms that rely on hard mathematical problems (they are in fact believed to be hard). The intractability of these problems are exploited to provide the security of cryptographic protocols. The problem of factoring large integers (e.g. RSA), solving discrete logarithms (e.g. ElGamal, DSA, DH), or solving elliptic curve discrete logarithms (e.g. ECDSA, ECDH) are the most prominent mathematical problems used for cryptography nowadays. However, there are also other primitives known where protocols and schemes are based upon. Such protocols are, for instance, the NTRU cryptosystem which is based on the Shortest Vector Problem in a Lattice or the Goldwasser-Micali cryptosystem that is based on the Quadratic Residuosity Problem.

4.2 Authentication Protocols based on Elliptic Curves

The fundamental security goal of authentication protocols is the resistance against impersonation through both passive and active attacks. Passive attacks extract information by passively monitoring (eavesdropping) multiple protocol executions. In active attacks, an adversary plays the role of the verifier and extracts information by actively interacting with the prover.

In the following, we focus on authentication protocols that are at least provably secure against passive attacks. They are constructed on basis of different types of schemes but they use the same cryptographic primitive which is based on the intractability of the elliptic curve discrete logarithm problem (ECDLP). In fact, elliptic curve cryptography (ECC) has gained much importance in the field of low-resource devices such as contactless smart cards and embedded systems. The main benefits of ECC compared to traditional cryptographic primitives like RSA are the significant improvements in terms of speed and memory. In fact, memory is one of the most expensive resources in the design of embedded systems which encourages the use of ECC on such platforms.

At first, we describe protocols providing entity authentication. These protocols are composed of encryption schemes, signature schemes, or identification schemes. Second, we describe message-authentication protocols that are constructed using signature schemes. All protocols involve two parties (a reader and a tag) and provide unilateral authentication.

The following notation is used throughout the rest of the chapter. Common parameters to all entities are: the underlying finite field $\mathbf{F}_q$, the elliptic-curve parameters a and b, the curve point P with order n, and the public-curve point Q. Finite elliptic-curve points are upper case such as P and Q. The private key is denoted by d, the protocol challenge is denoted by c, and the response is denoted by y. All used random numbers (ephemeral keys) are referred to as r or k and the security parameter t defines the number of bits for the challenge. The variable e represents the output of the one-way hash function h.

4.2.1 Entity Authentication through Encryption Schemes

One way to reach entity authentication is to demonstrate the knowledge of a private key through the decryption of a ciphertext. In view of ECC, this can be achieved by applying the ECC-based variant of the ElGamal encryption scheme [ElG84]. In Figure 4.2, the principle of an entity-authentication protocol is shown that is based on an encryption scheme. First, the reader generates a random number r and encrypts r (using the public key Q) together with the identifier ID of the tag. Then it sends the result c, the identifier ID, and the witness $e = h(r)$ of the random number r to the tag. The tag decrypts the message and verifies the obtained values. It sends the decrypted value y to the reader which accepts if it is equal to the previous generated random number r. Note that the random number is necessary to prevent replay attacks, the iden-

tifier avoids reflection attacks, and the witness e is used to preclude chosen-text attacks [MvOV97].

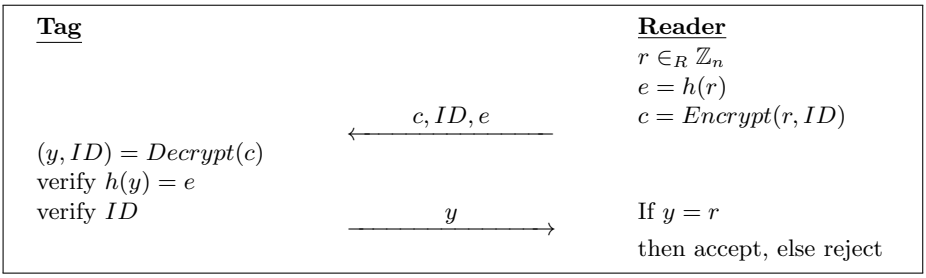


Figure 4.2: Entity authentication based on an encryption scheme.

Although encryption-based entity-authentication protocols are semantically secure and provide security against passive and active attacks, they lack several basic needs which often limit the use in practical applications. First, they rely on encryption algorithms which would not be licensed for an export to external countries. Second, they often make use of additional functionalities such as hash functions and timestamps. These building blocks are often more expensive in terms of implementation costs compared to other protocols providing the same cryptographic service. Thus, encryption-based schemes are usually avoided especially on resource-constrained devices like RFID tags.

In the following, we describe three entity-authentication protocols that are based on identification schemes. These schemes have been basically adapted to work also without encryption primitives and hash functions. They provide an interactive proof-of-knowledge and have become important for RFID applications because of their small footprint and light-weight structure. In addition, they provide the ability to pre-compute values to perform "on-the-fly" authentication [GPS06] which allows very fast authentication of RFID tags in the field.

4.2.2 Entity Authentication through Identification Schemes

As opposed to encryption schemes, where a proof is provided by applying a challenge-response protocol, most identification schemes give proofs that are probabilistic rather than absolute. A verifier (reader) is convinced by a prover (tag) to be in possession of a secret key (the private key). The described schemes consist of three communication passes that may be executed several times (sequential version) or only in a single round (parallel version).

The first protocol is given in Figure 4.3. It is an ECC-variant of the authentication protocol published by C. Schnorr in 1989 [Sch90]. First, the tag generates a random number r and calculates the elliptic-curve point X as a witness. The tag sends the witness to the reader. Second, the reader generates a challenge c and sends it to the tag. The tag now calculates y and sends it as a response back to the reader which accepts or rejects the tag. The protocol of Schnorr is an interactive identification scheme that provides the properties completeness,

soundness, and honest-verifier zero-knowledge. The latter property means that it provides the perfectly zero-knowledge property only when the tag interacts with a honest reader. For cheated readers, which may choose the challenge to be too large (super-polynomial), it loses the zero-knowledge property. The protocol is thus secure against passive adversaries under the elliptic-curve discrete logarithm assumption but it is not secure against active attacks [Sch90].

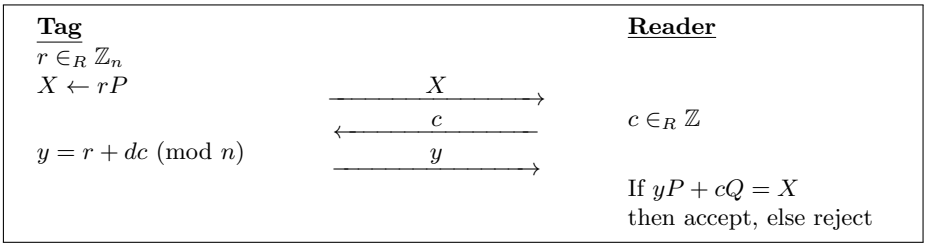


Figure 4.3: ECSchnorr authentication protocol.

The second authentication protocol was published by T. Okamoto [Oka93] in 1992. It is shown in Figure 4.4. Compared to the protocol of Schnorr, it provides additional security against active and concurrent attacks. First, the tag generates two random numbers r_1 and r_2. Using these two random numbers, it calculates the elliptic-curve point X and sends it to the reader. As a second step, the reader picks a challenge c and sends it to the tag. Then, the tag calculates two responses y_1 and y_2 and sends it back to the reader which verifies the authenticity of the tag. As opposed to the protocol of Schnorr, Okamoto provides a witness-indistinguishable proof-of-knowledge.

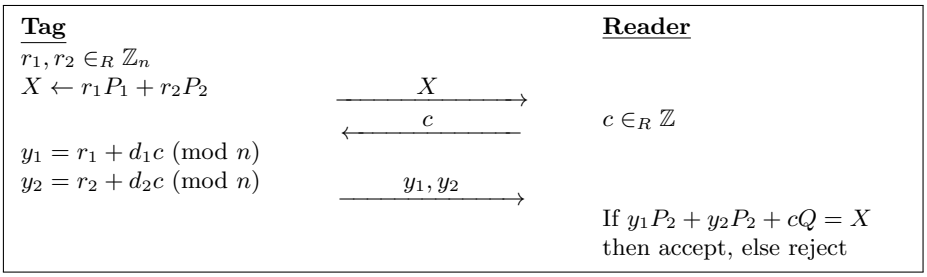


Figure 4.4: ECOkamoto authentication protocol.

The third authentication protocol (given in Figure 4.5) is an ECC-variant of an interactive identification protocol proposed by M. Girault, G. Poupard, and J. Stern in 2001. Due to the authors it is called GPS and was part of the European project NESSIE [P+03] and has been standardized in the ISO/IEC 9798-5 standard [Int04a] since 2004. The protocol is similar to Schnorr but eliminates the modular reduction during the response calculation by performing the operations in $\mathbb{Z}$. Like the Schnorr protocol, GPS is proven to have the (statistical) zero-knowledge property if the challenge c is not chosen too large. It provides

the honest-verifier zero-knowledge property and is thus only secure against active attacks under a given honest-reader assumption [GPS06].

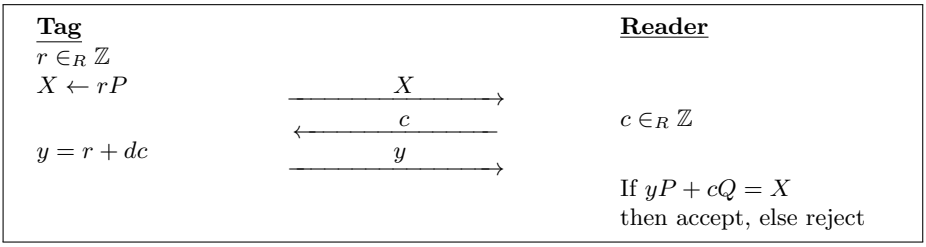


Figure 4.5: ECGPS authentication protocol.

Note that any of the interactive identification schemes can be transformed into a (non-interactive) signature scheme using the transformation technique proposed by A. Fiat and A. Shamir [FS87]. Therefore, the challenge c has to be replaced by the outcome of a cryptographic hash-function h. The input of the hash function is the witness x concatenated with the message m that has to be signed $c = h(x, m)$. For the security of the signature scheme it is necessary to apply a hash function that is collision resistant, i.e. it is hard to find two messages that lead to the same hash value. Furthermore, the size of the hash function must be chosen not too small to prevent existentially unforgeable signatures under adaptive chosen-message attacks. In the following, we present such a signature scheme that uses the SHA-1 hash function to generate digital signatures. Next to entity authentication it provides message authentication by signing messages of a verifier.

4.2.3 Authentication through Signature Schemes

A signature scheme can be used to provide both entity as well as message authentication. A challenge or message is signed by the prover that can be validated even at some later instant of time. Signature schemes provide therefore two additional cryptographic services which are non-repudiation and data integrity.

Figure 4.6 shows an entity authentication protocol based on the ECDSA signature scheme. The protocol has been standardized in ISO/IEC 9798-3 [Int93]. ECDSA is the elliptic-curve based variant of the digital signature algorithm (DSA). The DSA has been proposed in 1991 by the National Institute of Standards and Technology (NIST). ECDSA has been standardized by several organizations such as ANSI [Ame05], IEEE [IEE04], NIST [Nat00], and ISO/IEC [Int06].

The protocol is structured as follows. First, the reader generates a random challenge c_1 and sends it to the tag. The tag also generates a random number $k \in [1, n-1]$ that is often referred to as ephemeral key. Then, an elliptic-curve scalar multiplication is performed using k and a base point P. The resulting x-coordinate of the scalar multiplication $[k]P$ is taken modulo n and stored in variable r. After that, the tag hashes the challenge c_1 of the reader together with an own generated random number c_2 and the identifier of the reader. As

a hash function the SHA-1 algorithm [Nat08] is used. Next, the private key d is multiplied with the intermediate value r. The result is then added to the output of the hashed message $e = h(m)$. The final signature can then be calculated by inverting the ephemeral key k and multiplying it with the intermediate value $e + dr$ modulo n. It then consists of the tuple (r, s), which is sent to the reader.

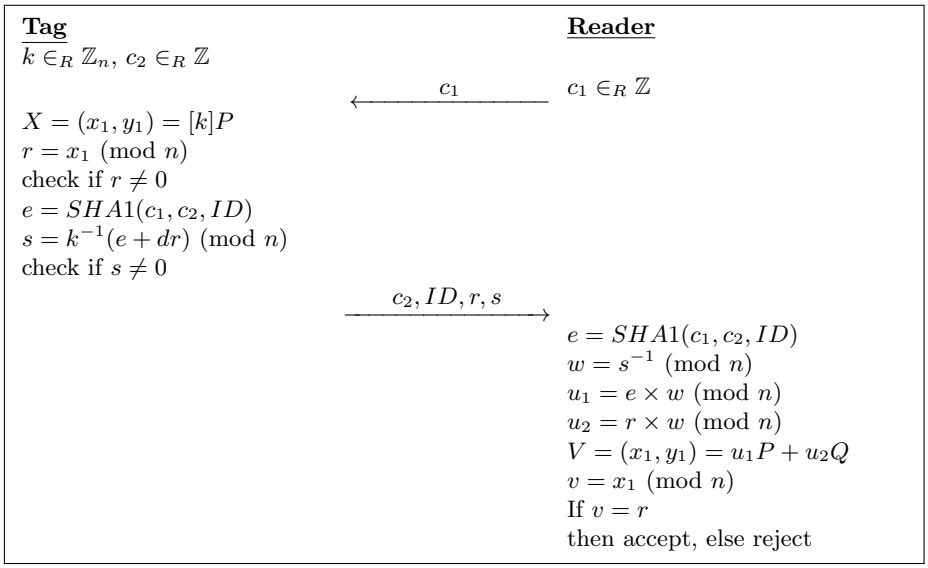


Figure 4.6: Entity authentication based on the ECDSA signature scheme.

4.3 Side-Channel Attacks on Public-Key Protocols

There exist many articles that present side-channel analysis attacks on elliptic curve-based algorithms. Most of the proposed side-channel attacks on ECC target the secret scalar during the scalar multiplication. In view of ECDSA, such an attack would reveal the ephemeral key used during the signature generation process. An adversary, which is in possession of the ephemeral key k, the message m (and thus the hashed message $e = h(m)$), and the resulting signature (r, s), is able to easily recover the (long-term secret) private key d by calculating

$$d = r^{-1}(ks - e) \pmod n. \tag{4.1}$$

In the following, we describe a side-channel attack that reveals the private key of ECDSA. Instead of attacking the ephemeral key k of ECDSA, we target the private-key multiplication during the signing process. The given attack can be generalized on other public-key based protocols that involve a multiplication

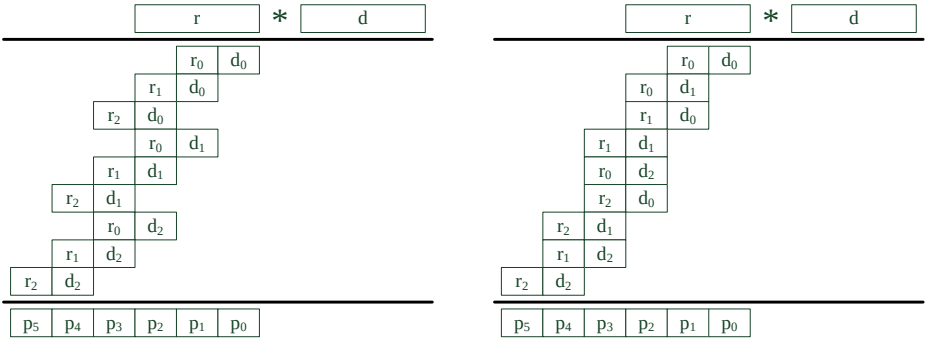


Figure 4.7: Operand scanning form (left) and product scanning form (right)

of the private key and a known intermediate value. Note that most protocols in practice are vulnerable to this attack.

4.4 Description of Our Attack

We present an attack on an entity-authentication protocol that uses ECDSA as a signature scheme. The target of the attack is an intermediate value that depends on the private key on the one hand and on some known value on the other hand. Regarding the ECDSA signature generation algorithm shown in Figure 6.12, the private key d is multiplied with the output of the scalar multiplication r. The private key is static and the output of the scalar multiplication is random since the ephemeral key k is chosen randomly for each signature generation. Furthermore, r is publicly known because it is part of the signature. In the light of these facts, we are able to perform an attack on intermediate values that are processed during the calculation of the private-key multiplication dr.

Common hardware implementations use multi-precision arithmetic for finite-field multiplication. One of the most used algorithms are the operand scanning and the product scanning (Comba) multiplication methods which are depicted in Figure 4.7. Both algorithms multiply the words of two n-word operands. In our case, those are r_i (the input) and d_j (the key). The resulting partial products are then added to a cumulative sum p. This results in n^2 partial products. Note that one word of the private key is processed n times during the whole multiplication.

What seems obvious at first glance turns out to be more complex in practice. The finite-field multiplication is a linear function that multiplies a constant value with a random input value. That means that shifted bit combinations of a key word have a linear impact to the multiplication result. When the key is shifted x times, the result is also shifted x times. Therefore, it is evident that in power-analysis attacks, one or more correlation peaks occur for only one key word. This is due to the fact that all bit combinations of the key word will result in

the same Hamming-weight[1] value of the multiplication output. The number of possible shifts s of the key word d_i can be calculated as follows:

$$s(d_i, l) = log_2(gcd(d_i, 2^l)) + l - \lfloor log_2(d_i) + 1 \rfloor, \tag{4.2}$$

where l represents the word size. Note that the maximum number of key shifts is equal to the word size, i.e. 16 for a device using 16 bit operands (in this case the key word has a Hamming weight of one and the value of the shifted key combinations are a multiple of 2^x where $x = 0..15$). Due to these facts, the decision of which hypothetical key is the correct one and which are incorrect keys seems therefore infeasible. It is clear that this makes an attack much more inefficient compared to attacks on intermediate values that occur after non-linear functions such as the S-box in DES [Nat99] or AES [Nat01].

Our attack can be separated into two steps. In the first step, we target the output of all partial products and perform an attack on that intermediate value. For each partial product, we obtain one or more promising key candidates due to the reasons described above. For a device with a 16-bit word size, for example, we obtain up to 16 promising key candidates. Hence, we get up to 16 key candidates for each private-key word d_i. In the second step, we target the output of the final multiplication product p. Each word of this product depends on one or more different key words. Thus, we can use the information obtained from the first step and use all obtained key candidates d_i to perform an attack on the final product words p_i. After revealing the key candidates for d_0 and d_1, we can attack the second product word p_1 to obtain the correct key word d_0. Incorrect key hypotheses will show low correlation peaks so that they can be eliminated from the correct key hypothesis that causes a higher correlation. By following this way, an attack that targets each of these product words p_i will yield all private-key words d_i successively.

4.5 Related Work

As already stated before, most side-channel attacks on ECC-based public-key implementations have been performed on the secret scalar during scalar multiplication. S. Coron [Cor99] was one of the first, who showed how to extract the secret scalar of a scalar multiplication using SPA attacks. He showed that the left-to-right and right-to-left binary scalar multiplication (often referred as to *double-and-add* method) is highly susceptible to such attacks. By simply inspecting one measured power trace of a scalar-multiplication implementation, a difference in the power consumption can be easily observed depending on whether a doubling or an addition operation is performed. This has its reason in the fact that a point doubling operation is only performed when the current bit of the secret scalar is 1. It is however not performed when the bit of the

[1] the Hamming weight power model is often used in practice and is further used in order to describe the attack

scalar is 0. As a countermeasure against this attack, Coron proposed to perform a point addition in every loop iteration of the scalar multiplication. The so-called *double-and-add-always* multiplication provides resistance against SPA attacks. Nevertheless, it provides no security against fault attacks like safe-error attacks due to the lack of a regular structure which is described and discussed in Chapter 5.

C. Gebotys et al. [GG03] performed an SPA attack on a 192-bit ECC implementation running on a DSP VLIW processor in 2003. They measured 60 power traces and analyzed the ECC group operations due to side-channel leakage. The difference between a point addition and doubling could be easily distinguished using a single power trace.

B. Örs et al. [OOP03] made similar observations by performing an SPA attack on the double-and-add scalar multiplication method running on an FPGA prototyping platform. They clearly discerned differences in the power-consumption trace for point addition and doubling. They were able to successfully reveal the 160-bit secret scalar from one power trace.

M. Medwed et al. [Med07] presented a template-based SPA attack in 2007 which extracts the ephemeral key of an ECDSA implementation. They performed an SPA attack on a 32-bit ARM7 microprocessor using side-channel templates [CRR03]. They created templates for several intermediate values and performed a matching phase which reveals the ephemeral key bit by bit having a success probability of almost 1.

For a comprehensive overview of state-of-the-art attacks and countermeasures on ECC see the work of J. Fan et al. [FGM+10].

Figure 4.8: A passively powered RFID-tag prototype that is capable of generating digital signatures using ECDSA.

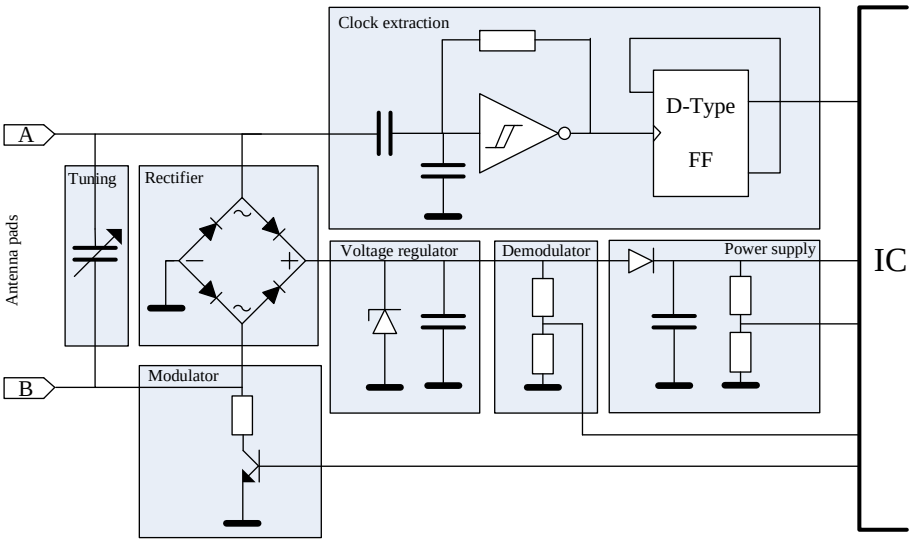


Figure 4.9: Schematic of the analog front-end of our passively powered RFID-tag prototype.

4.6 A Passive ECDSA-Enabled RFID-Tag Prototype

In this section, we present the design and implementation of a passively powered RFID-tag prototype that has been used throughout our experiments. A picture of the prototype is shown in Figure 4.8. It consists of an antenna, an analog front-end, and a low-power digital controller. The antenna has four windings and has been designed according to ISO 7816 [Int89]. The antenna is connected to an analog front-end that transforms the received analog signals of the reader to the digital world of the digital controller. The controller includes a digital RFID front-end and a low-power hardware implementation of ECDSA. In the following, we describe the analog front-end and the digital controller in more detail.

4.6.1 The Analog Front-End

The analog front-end is composed of the seven parts shown in the schematic view in Figure 4.9. In the first stage, the antenna is connected to a matching circuit which tunes the antenna to the 13.56 MHz carrier frequency of the reader. After that, a bridge rectifier has been assembled using low-voltage drop schottky diodes. The rectified signal is then smoothed and fed into a slow envelope detector to provide a stable power supply for the digital controller.

In order to comply with many commercial RFID tags, we designed a clock extraction circuit that is able to regenerate a system clock out of the 13.56 MHz

reader signal. For this, a relaxation oscillator has been implemented using an inverting Schmitt trigger, one resistor, and a capacitor which produce a stable 13.56 MHz square-wave clock. The clock is then divided by two using a D-type flip-flop to provide a 6.78 MHz clock frequency that is needed for the controller.

For receiving and sending of data, both a modulation and a demodulation circuit have been integrated. For data modulation, a resistor is used that can be connected and disconnected to the antenna by the controller. After switching the resistor, a significant amount of additional power is drawn out of the reader field. This so-called load modulation is then detected and demodulated by the reader.

4.6.2 The Digital ECDSA-Enabled RFID Controller

The digital controller is an elliptic-curve point multiplication device with an ISO 15693 [Int01b] compatible digital RFID front-end. It is capable of computing the multiplication of a scalar value with a point on the NIST standardized elliptic curve B-163 [Nat00]. The B-163 curve is defined on the binary extension field $\mathbb{F}_{2^{163}}$. The controller was fabricated on the UMC L180 GII 1P/6M 1.8V/3.3V CMOS process. The controller has a total area of 15 630 Gate Equivalents (GE)

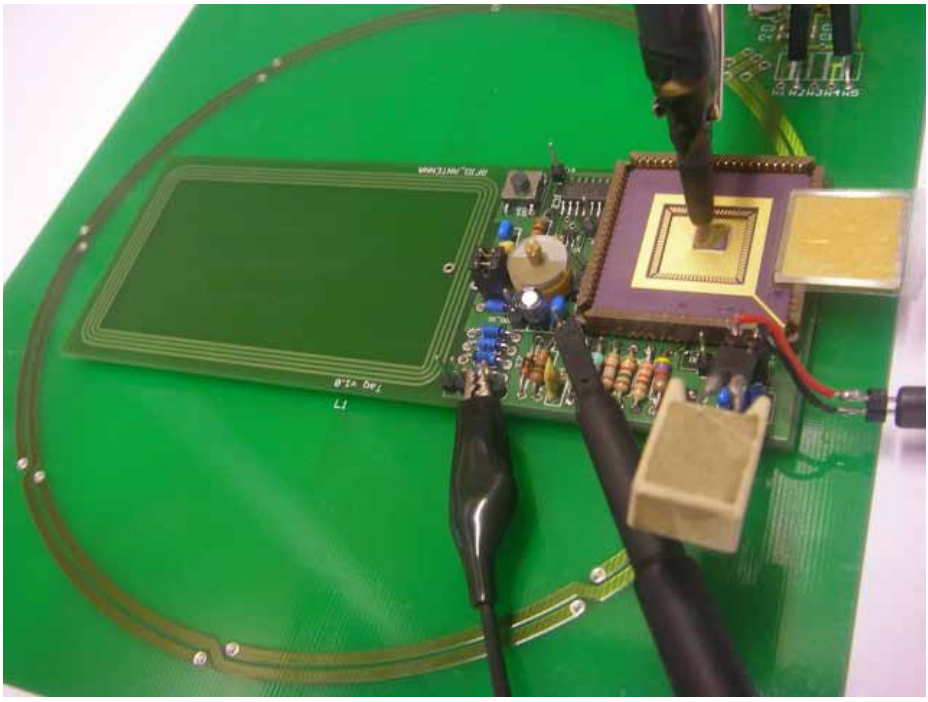


Figure 4.10: RFID measurement setup involving our tag prototype lying on the reader antenna.

while an overhead of 654 GE is needed by components for production testing. The digital RFID front-end requires 1 726 GE and the ECC core 13 250 GE. This includes 1 346 GE for a memory slot to enable the separate setting of the ephemeral key k.

The controller must be operated at a fixed frequency of 6.78 MHz. This is half of the carrier frequency. Internally, this frequency supplies two different clock domains. One of them is used for the RFID interface and has a frequency of 106 kHz. The other one clocks the ECC core at 847.5 kHz. The whole chip has an estimated power consumption of about 176 µW [HWF08a, HWF08b].

4.7 The Measurement Setup

The measurement setup is composed of several parts. We used a PC, an RFID reader, the RFID-tag prototype, a digital sampling oscilloscope (DSO), a differential probe, and a near-field measurement probe. The PC controls the overall measurement process. It is connected to the DSO and the RFID reader. We used an 8-bit oscilloscope that offers an acquisition bandwidth of up to 1 GHz. As a reference measurement, an active differential probe has been connected in parallel to a 1 Ω resistor that is placed in series to the *VDD* core power supply. For electromagnetic measurements, we used a tiny magnetic near-field probe that allows the sensing of signals only up to a few millimeters. This already reduces the noisy reader signal in an early stage of the acquisition process. The sensed signals are then amplified by a 30 dB pre-amplifier before they are sampled by the oscilloscope. The sampling rate has been set to 1 GS/s for all measurements. Figure 4.10 shows the RFID measurement setup involving our tag prototype that lies on the antenna of a reader.

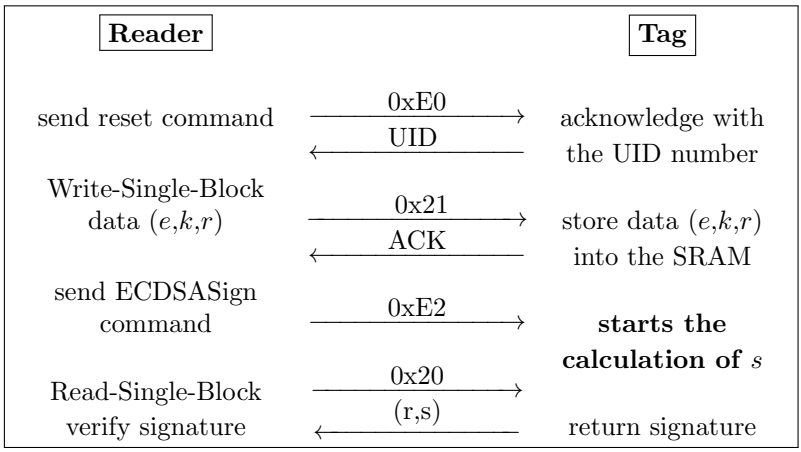


Figure 4.11: RF communication between the reader and the tag.

We have used a standard RFID reader that supports mandatory and optional ISO 15693 commands such as *Inventory* or *Select*. It is also able to send custom

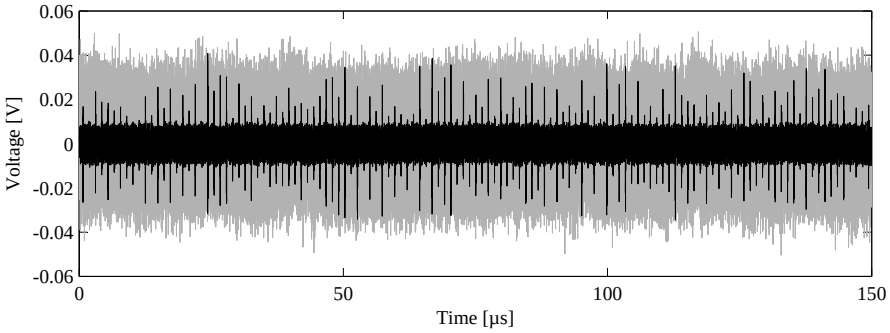


Figure 4.12: Power trace (black) and (30-times magnified) EM trace (gray) during the calculation of the private-key multiplication.

commands that are needed to start the ECDSA signature generation. We defined three custom commands. The first command (0xE0) performs a hardware reset and loads initial data (like the base point) from Read Only Memory (ROM) into the internal Static Random Access Memory (SRAM) of the tag controller. The second command (0xE1) starts the scalar multiplication and the third command (0xE2) calculates the signature $s = k^{-1}(e + dr)$.

The communication flow between the reader and our tag prototype is shown in Figure 4.11. First, a reset command (0xE0) is sent to the tag. The tag responds with its unique ID (UID) number. Instead of calculating the scalar multiplication in each power trace acquisition, we pre-calculated the value r and loaded it into the SRAM before starting the power acquisition of the finite-field multiplication. This is done by sending the Write-Single-Block (WSB) command (0x21) of ISO 15693. After that, the reader sends the ECDSASign command (0xE2) to start the calculation of the digital signature (r, s). As a trigger signal, the oscilloscope was set to listen on the End-of-Frame (EOF) sequence of the Write-Single-Block command.

The RFID controller inverts the ephemeral key k in about 11.8 ms. The private-key multiplication needs about 150 µs, the hash-value addition and the final multiplication need around 600 µs. In our setup, the power consumption as well as the electromagnetic emanation of our device were acquired simultaneously throughout the private-key multiplication. Figure 4.12 shows one measured power trace (drawn in black) and a 30-times magnified electromagnetic trace (drawn in gray).

4.7.1 Pre-Processing RFID Power Traces

Measurements in RFID environments are very noisy due to the high signal of the reader device. We therefore applied two pre-processing techniques: trace alignment and trace decimation.

First, we aligned all measured traces in both horizontal and vertical orien-

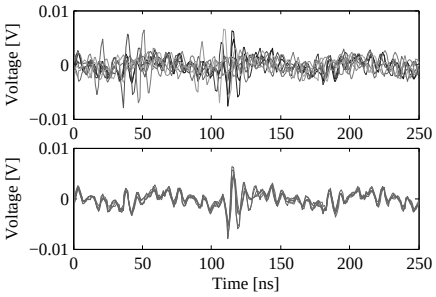


Figure 4.13: Misaligned traces (upper plot) and aligned traces (lower plot) of electromagnetic measurements.

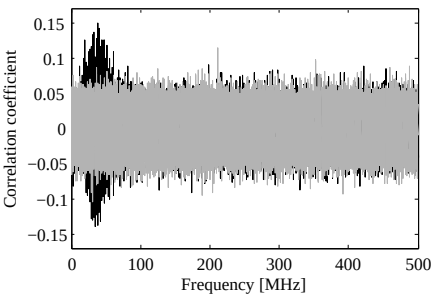


Figure 4.14: Correct (black) and incorrect (gray) correlation traces of the frequency-based DPA attack using 2 000 power traces.

tation. We determined a specific trace pattern which was used to align the remaining traces using the Least-Mean-Square (LMS) algorithm. The misaligned traces are shown in the upper plot of Figure 4.13. The lower plot demonstrates the traces after alignment. Our experiments have shown that without alignment or even poor alignment, successful attacks become largely infeasible due to the high noise of the measurement setup.

Second, we applied a trace-decimation technique that has been proven to be very useful throughout our experiments. Decimation is a technique typically used in signal processing. It performs two actions: filtering and re-sampling. First, an appropriate low-pass filter is applied to the measured power traces. This filter attenuates all frequency signals above a certain cut-off frequency. Second, it re-samples the smoothed traces at a lower rate. Decimation has therefore two major advantages. On the one hand, misalignment are compensated due to the averaging of filtering. On the other hand, all measured traces become shorted in their length. Both properties are a major concern for successful attacks in RFID environments.

In order to apply the decimation technique to our power traces, we determined a proper cut-off frequency. This frequency has to be chosen in a way so that signals are eliminated that do not carry data-dependent information. D. Agrawal et al. [AARR03] have shown that there exist data-dependent information in the lower frequency spectrum. The higher the frequency, the weaker will be the signals that carry interesting information. Due to this fact, we have performed a frequency-based DPA attack that was first introduced by C. Gebotys et al. [GHT05] in 2005. All measured power traces are transformed into the frequency domain using the Fast Fourier Transformation (FFT). Instead of correlating the sample points in the time domain, the sample points are correlated in the frequency domain. As a target of the attack, we have chosen the same intermediate value used in the attack described in Section 4.4. Figure 4.14 shows the result of the attack using 2 000 power traces. The correct key hypothesis is

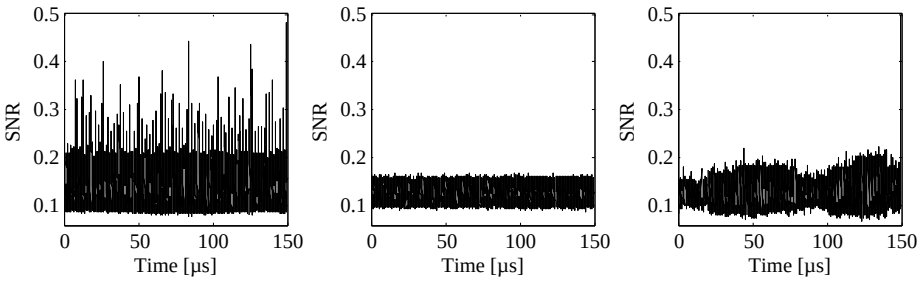


Figure 4.15: SNR of the power traces (left), SNR of the EM traces without pre-processing (middle), and SNR of the pre-processed EM traces (right)

drawn in black and the incorrect key hypotheses are drawn in gray. It can be clearly seen that there is a high correlation below 50 MHz. Above this frequency, no significant correlation can be discerned. On this account, we applied an 8th-order Chebyshev (Type 1) low-pass filter with a cutoff frequency at 50 MHz and down-sampled the traces accordingly. All traces have been decimated from 300 000 sampling points to only 32 500 samples.

4.7.2 Device Characterization and Pre-Processing Evaluation

Next, we characterize our tag prototype concerning side-channel information leakage. First, the noise of the measurement setup is characterized. Second, the data-dependent signal that is leaked by the device is determined. After that, we calculate the signal-to-noise (SNR) ratio of the power and the electromagnetic measurements and compare them. Furthermore, we evaluate the pre-processing techniques by comparing measurements with and without trace alignment and trace decimation.

The noise of the measurement setup has been characterized by calculating the mean of all traces that were captured by processing constant data. This avoids data-dependent power variations and allows the determination of the measurement noise. Data-dependent signals, in contrast, have been characterized within two steps: First, the mean of those traces that process the same input data is calculated. Second, the variance of these mean traces is calculated. The SNR can now be calculated by dividing the variance of the obtained mean-signal trace from the variance of the calculated noise trace [MOP07]. For the SNR calculation, 2 000 traces have been used.

Figure 4.15 shows the result of the characterization and performance evaluation. The left plot in the figure shows the SNR of the power traces. A maximum SNR of 0.45 has been obtained. In the middle plot of the figure, the result of the EM traces is given which have not been pre-processed. It can be seen that the signal components are below the noise floor. With this number of traces, an attack would fail due to the low SNR. The right plot of the figure shows the

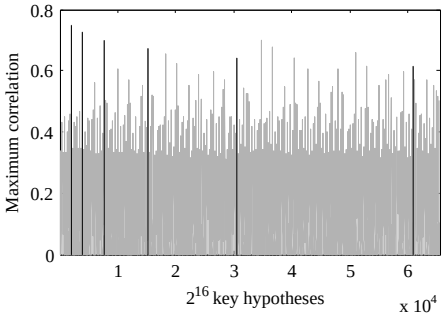


Figure 4.16: Maximum correlation coefficient of all 2^{16} key hypotheses for the first private-key word d_0 using 2 000 power traces.

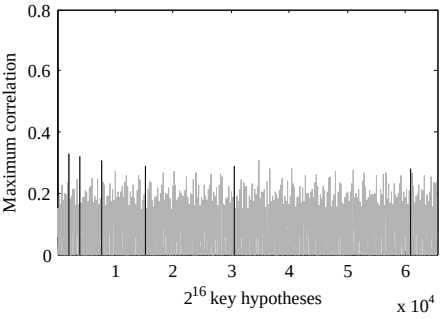


Figure 4.17: Maximum correlation coefficient of all 2^{16} key hypotheses for the first private-key word d_0 using 2 000 EM traces.

SNR of the pre-processed EM traces. Due to the pre-processing, the SNR could be significantly increased to a maximum of 0.22. The signal components are much weaker as compared to the power traces but an attack will still succeed as shown in the next section.

4.8 Results

This section presents results of power and electromagnetic (EM) analysis attacks on our ECDSA-enabled RFID-tag prototype. First, we perform a reference attack using a contact-based power analysis. The power consumption of the RFID-tag prototype is measured over a resistor that has been placed in series to the integrated RFID controller and the analog front-end. Second, we perform a contact-less attack using EM analysis by using a magnetic near-field probe. In both scenarios, the tag was powered passively by the field.

The first attack targets the first partial product of the multi-precision multiplication unit of our RFID controller. The controller implements a 16-bit Comba-multiplication unit so that we have to test 2^{16} key hypotheses that are multiplied with the known intermediate value r. The target has been the 32-bit output of the multiplication. However, our experiments have shown that individual bits of our device leak more or less than others. Hence, we have modeled the power consumption by weighting the Hamming weight of the higher 16 bits and the lower 16 bits differently to obtain the highest correlation.

Figure 4.16 and Figure 4.17 show the result of the power and EM attack. For both attacks, 2 000 traces have been used. The x-axis represents all possible key hypotheses and the y-axis represents the maximum absolute correlation of each resulting correlation trace. It is clearly discernable that the results obtained from the EM traces reach only half the correlation value as they have been obtained from the power traces. The reason for this is the lower SNR of the

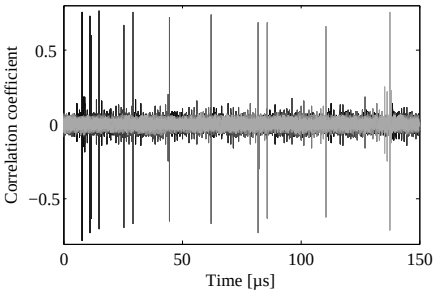


Figure 4.18: Correlation traces of all partial products $r_i * d_0$ using 2 000 power traces.

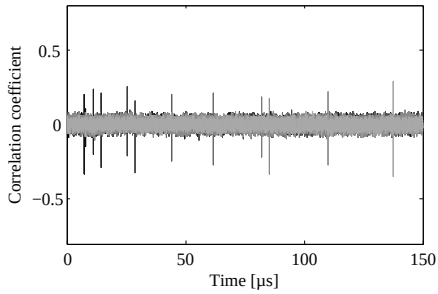


Figure 4.19: Correlation traces of all partial products $r_i * d_0$ using 2 000 EM traces.

EM measurement calculated in the previous section. Furthermore, it can be observed that the highest peak has been obtained for the key word 1901 and reached a correlation coefficient of 0.75 for the power traces and 0.33 for the EM traces. However, there exist also five other key hypotheses which result in a high correlation[2]. These are 3 802, 7 604, 15 208, 30 416, and 60 832 (marked as black lines in the figures). Obviously, these values have the same bit representation as 1 901 but are gradually shifted to the left. This is due to the fact that finite-field multiplication is a linear function where shifted bit combinations of the correct key have a linear impact to the multiplication result.

Next, we performed the same attacks on all other partial products that involve the first private-key word d_0. Figure 4.18 and Figure 4.19 show the result of the attacks. The correlation results of the correct key-hypothesis d_0 of all partial products are plotted on top of each other. Eleven peaks are observable that occur at locations in time when the output of the partial products is stored into the internal registers of the controller. Due to the structure of the Comba multiplication unit, the distance between these results becomes larger the more partial products are calculated. The first partial product is calculated after about 10 μs and the last one after about 140 μs. The power-analysis attacks lead to a mean correlation of 0.72. The EM attacks yielded a mean correlation of 0.22.

After revealing the promising key candidates of the first private-key word d_0, we performed attacks on all partial products that involve the second private-key word d_1. The attacks led us to two promising key candidates: 24 027 and 48 054. Then, we performed an attack on the second result of the final multiplication product p_1. This product word involves the calculation of the first and the second private-key word. Thus, we have to test 12 promising key hypotheses. A high correlation will occur when both hypotheses are correct. Incorrect hypotheses will show no peak. In Figure 4.20, the result of the power-analysis attack is given using 2 000 power traces. The correct key hypotheses (drawn in black) 1 901 for

[2] the peaks do not have the same correlation value since our power model weighted the lower and higher bits of the 32-bit multiplication output differently.

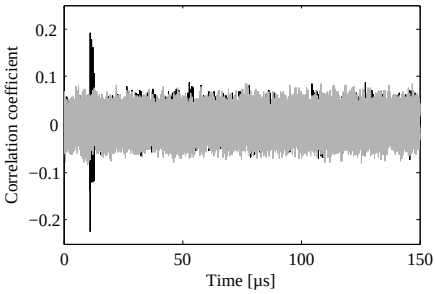


Figure 4.20: Result of the power-analysis attack on the final multiplication product p_1 using 2 000 traces.

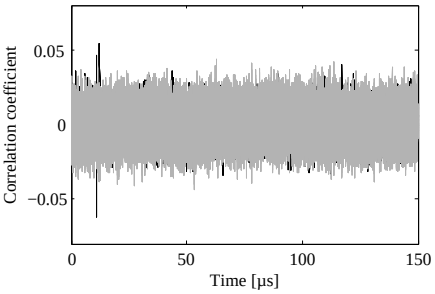


Figure 4.21: Result of the EM attack on the final multiplication product p_1 using 10 000 traces.

d_0 and 48 054 for d_1 yield a high correlation while all other key hypotheses (drawn in gray) show no peak in time when the final product is stored into the internal register of the controller. Figure 4.21 shows the result of the EM attack using 10 000 traces. It provides a much smaller correlation as compared to the result of the power-analysis attack. Nevertheless, the correct key can be easily discovered from the incorrect ones.

All other private-key words have been extracted by following the same strategy. Both power and electromagnetic attacks have been successful. The attacks revealed the entire private key of the ECDSA implementation which enables us to forge digital signatures and therefore to impersonate any entity and person by cloning the extracted key.

4.8.1 Countermeasures

There exist several countermeasures against DPA attacks on ECC-based protocols. Most of the countermeasures target the protection of the secret scalar during scalar multiplication. S. Coron [Cor99] proposed three countermeasures which are often referred in literature. The first countermeasure randomizes the private exponent d by calculating $d' = d + k$, where k is a random number of size n (e.g. 20 bits). This changes the intermediate values during scalar multiplication $Q = d'P$ in every execution and thus makes DPA attacks ineffective.

The second countermeasure is based on blinding the point P. This is done by adding a random point R. The scalar multiplication $Q = dP$ is therefore replaced by $Q = d(R + P)$. After that computation, the point $S = dR$ is subtracted from Q to get $Q = dP = d(R + P) - S = d(R + P) - dR$.

As a third countermeasure, Coron proposed to randomize the projective coordinates of the involved point P (often referred to as Randomized Projective Coordinates (RPC) countermeasure). Thus, the point $P = (X, Y, Z)$ is randomized by a random scalar λ to $P' = (X\lambda, Y\lambda, Z\lambda)$. The random value λ is then reduced by transforming the projective coordinates back to affine coordinates.

Note that the security of the countermeasures largely depends on the practical implementation. K. Okeya et al. [OS00] emphasized that wrong implementations would lead to an insufficient protection against power-analysis attacks. They also discussed the proposed countermeasures of Coron and claimed that the first two proposals fail if they are implemented wrong. L. Goubin [Gou03] showed that protected implementations of scalar multiplications can be even broken if an adversary can chose the base point dynamically. The so-called Refined Power Analysis (RPA) attack exploits the fact that one coordinate of the involved base point contains a zero value. This would lead to side-channel leakages which can be exploited by attacks. T. Akishita et al. [AT03] proposed so-called Zero-Value Point Attacks (ZPA) as an extension of the RPA attacks presented by Goubin. They observed that even if the input coordinate does not provide a zero value, intermediate registers might get zero values during computation. They found several points P even for recommended elliptic curves that provide such zero-values. However, the proposed attacks assume that the base point P can be chosen by the adversary and the secret scalar is fixed during the computation which makes attacks on implementations of ECDSA signature generation ineffective.

To protect implementations against attacks that target the modular multiplication algorithm of public-key protocols, one can make use of secure CMOS logic styles or apply masking or hiding techniques [MOP07]. In case of ECDSA, the final signing process can be modified to blind the private key with a random number, i.e. a multiplication with the unknown private key with an unknown random number. This would avoid first-order power-analysis attacks. Such a countermeasure is very cheap in has been integrated in the ECDSA implementation described in Chapter 6.

4.9 Summary and Conclusions

In this chapter, we presented results of side-channel attacks on a public-key enabled RFID device. First, we introduced into public-key cryptography and define a general model for different public-key techniques. Second, we performed power and electromagnetic attacks on an RFID-tag prototype that implements a public-key authentication protocol using ECDSA. The prototype is able to be powered passively by the field of a reader. It includes a low-power hardware ECDSA implementation fabricated in 180 nm CMOS technology. The implementation is able to authenticate a tag to a reader by generating digital signatures.

As opposed to other proposed attacks that try to reveal the ephemeral key of ECDSA, we attacked the private key directly that is used during the calculation of the digital signature. Adversaries who are in possession of the private key are in fact able to forge signatures and thus can provide a valid proof of origin of RFID-labeled products. It is therefore mandatory to provide appropriate countermeasures against these attacks if such implementations are applied in security-related applications.

As an improvement for side-channel analyses on RFID devices, we further

described a useful pre-processing technique by decimating the measured side-channel traces. We performed DPA and DEMA attacks on our prototyping platform and successfully revealed the private key with less than 2 000 power traces and 10 000 electromagnetic traces.

We conclude that public-key enabled RFID devices are vulnerable to side-channel attacks. If the private-key gets multiplied with a known random value within a multi-precision multiplication, the implementation is vulnerable to the presented attack. It is not sufficient to protect only the scalar multiplication of ECC-based public-key protocol implementations.

Part II

Secure and Efficient Implementation of RFID Tags

5

Out-of-Place Formulæ for Elliptic Curve Cryptography in Co-Z Coordinate Representation

It is possible to write endlessly on elliptic curves (This is not a threat.)
— Serge Lang

In this chapter, we investigate new formulæ that provide very efficient computations on elliptic curves over prime fields. In particular, we present new formulæ for differential point addition and doubling using the Montgomery powering ladder as scalar multiplication method. The formulæ are based on a homogeneous projective coordinate representation that use a shared Z coordinate to increase the performance of ECC. In addition, we consider the practical constraint usually imposed by implementations of both modular multiplication and squaring which may not allow the result to overwrite any of the operands. All given formulæ make therefore use of *out-of-place* operations which assure that no additional memory is needed to perform finite-field arithmetic computations. The results of this chapter constitute an improvement of the state-of-the-art in designing efficient implementations of elliptic curve cryptography especially suitable for low-resource devices and embedded systems.

The results presented are outcomes of a joint work together with Marc Joye and Yannick Sierra. All authors have made significant contributions that can not be clearly separated.

First, we briefly introduce elliptic curve cryptography. We describe different scalar-multiplication methods on elliptic curves including the Montgomery lad-

der. Afterwards, we present new formulæ for (differential) addition-and-doubling and projective coordinate recovery in the co-Z coordinate representation system. Next we will discuss the difference between *in-place* versus *out-of-place* formulæ for ECC implementations. Finally, the results of our investigations are discussed in terms of security and performance. We show that the results outperform existing solutions in view of memory and speed.

5.1 Preliminaries

This section introduces some elementary background on elliptic curves. We refer the reader to [HMV04] for further details.

An elliptic curve E over a finite field $\mathbb{F}_q$ of characteristic $\neq 2, 3$ can be defined by the short Weierstraß equation

$$E : y^2 = x^3 + ax + b\,,$$

where $a, b \in \mathbb{F}_q$ are curve parameters satisfying $4a^3+27b^2 \neq 0$ and $(x, y) \in \mathbb{F}_q \times \mathbb{F}_q$ represents a point on the elliptic curve. The set of all points on the elliptic curve together with the point at infinity $\boldsymbol{O}$ is denoted by $E(\mathbb{F}_q)$. It forms an (additively written) abelian group with the point at infinity $\boldsymbol{O}$ as the identity element.

Scalar multiplication

The main operation in elliptic curve cryptography (ECC) is the scalar multiplication, $\boldsymbol{Q} = k\boldsymbol{P}$, where $\boldsymbol{P}$ and $\boldsymbol{Q}$ are points on the curve E and k is a scalar such that $0 \leq k < \mathrm{ord}_E(\boldsymbol{P})$. The security of ECC primitives relies on the intractability to solve the elliptic curve discrete logarithm problem (ECDLP), i.e. determining k from $\boldsymbol{P}$ and $\boldsymbol{Q}$.

Point representation

The scalar multiplication uses two basic operations that are addition and doubling of points. The points can be represented in several coordinate systems. Points in affine coordinates are represented by two coordinates x and y but involve the computation of inversions in $\mathbb{F}_q$ which are relatively expensive operations. Due to this reason, most implementations represent the points in projective coordinates. In homogeneous projective coordinates, each affine point (x, y) is represented by three coordinates (X, Y, Z) where $x = X/Z$ and $y = Y/Z$. Another coordinate system that is widely used in practice is the Jacobian projective coordinate system. There, the relation $x = X/Z^2$ and $y = Y/Z^3$ is used to represent the points. The curve equation in Jacobian coordinates becomes $E : Y^2 = X^3 + aXZ^4 + bZ^6$.

Point addition

Let $\boldsymbol{P_1} = (X_1, Y_1, Z_1)$ and $\boldsymbol{P_2} = (X_2, Y_2, 1)$ be two points represented in Jacobian projective coordinates on the curve. Then the sum $\boldsymbol{P_1} + \boldsymbol{P_2} = (X_3, Y_3, Z_3)$ (also

known as *mixed* sum since $Z_2 = 1$), is given by

$$\begin{cases} X_3 = (Y_2 {Z_1}^3 - Y_1)^2 - (X_2 {Z_1}^2 - X_1)^2 (X_1 + X_2 Z_1^2) \\ Y_3 = (Y_2 {Z_1}^3 - Y_1)(X_1 (X_2 {Z_1}^2 - X_1)^2 - X_3) - Y_1 (X_2 {Z_1}^2 - X_1)^3 \\ Z_3 = (X_2 {Z_1}^2 - X_1) Z_1 \end{cases} . \quad (5.1)$$

The formula for point doubling, $2\boldsymbol{P_1} = (X_4, Y_4, Z_4)$, is given by

$$\begin{cases} X_4 = (3{X_1}^2 + a{Z_1}^4)^2 - 8X_1 {Y_1}^2 \\ Y_4 = (3{X_1}^2 + a{Z_1}^4)(4X_1 {Y_1}^2 - X_3) - 8{Y_1}^4 \\ Z_4 = 2Y_1 Z_1 \end{cases} . \quad (5.2)$$

To evaluate the costs of the given formulæ we denote by M the cost of a field multiplication and by S the cost of a field squaring. For multiplications with fixed parameters such as the curve parameters, we use the notation $\mathsf{M}_\star$ (e.g. M_a, M_b). Additions and subtractions are later assumed to have the same complexity and are represented by add.

Evaluating formulæ (5.1) and (5.2) in terms of computational cost shows that a point addition needs $7\mathsf{M} + 4\mathsf{S}$ if $Z_2 = 1$ [HMV04]. Point doubling can be performed with $4\mathsf{M} + 4\mathsf{S}$ or $1\mathsf{M} + 8\mathsf{S} + 1\mathsf{M}_a$. For comparability reasons, we use the same performance metric as in the dedicated website Explicit Formulas Database (EFD) [BL].

Co-Z arithmetic

In 2007, Meloni proposed new point addition and doubling formulæ in Jacobian coordinates where the two involved points share the same Z-coordinate [Mel06]. We refer to this coordinate system as the *co-Z coordinate system*. When the two points satisfy this condition, the addition of two points can be evaluated much faster than an addition in Jacobian coordinates (actually even faster than a doubling operation in Jacobian coordinates). Let $\boldsymbol{P_1} = (X_1, Y_1, Z)$ and $\boldsymbol{P_2} = (X_2, Y_2, Z)$ the two points that share the same Z-coordinate, then the sum of the two points, $\boldsymbol{P_1} + \boldsymbol{P_2} = \boldsymbol{P_3} = (X_3, Y_3, Z_3)$, is given by

$$\begin{cases} X_3 = (Y_2 - Y_1)^2 - X_2 (X_2 - X_1)^2 - X_1 (X_2 - X_1)^2 \\ Y_3 = (Y_2 - Y_1)[X_1 (X_2 - X_1)^2 - X_3] - Y_1 (X_2 - X_1)^3 \\ Z_3 = Z(X_2 - X_1) \end{cases} . \quad (5.3)$$

This addition requires only $5\mathsf{M} + 2\mathsf{S}$. As observed in [Mel06], the given formulæ have the advantage of providing an equivalent representation $\boldsymbol{P_1'}$ of the point $\boldsymbol{P_1} = (X_1, Y_1, Z)$ such that the points $\boldsymbol{P1'}$ and $\boldsymbol{P3}$ have the same Z-coordinate value. Namely $\boldsymbol{P_1'} = (X_1 \lambda^2, Y_1 \lambda^3, Z\lambda)$ with $\lambda = (X_2 - X_1)$, is calculated without any additional cost since the coordinates are already computed as intermediate values in the addition formula (cf. Eq. (5.3)).

5.2 Scalar Multiplication Methods

There exist several algorithms to perform the scalar multiplication.

One of the most common methods is the *double-and-add* algorithm (a.k.a. left-to-right binary method), shown in Algorithm 1. It takes the binary representation of the scalar k as an input and processes the bits from left to right. A point doubling operation is performed at every iteration whereas point addition is only performed if the bit value, k_i, is 1.

Algorithm 1 Double-and-add

Require: $\boldsymbol{P} \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}$
Ensure: $\boldsymbol{Q} = k\boldsymbol{P}$
1: $\boldsymbol{R_0} \leftarrow \boldsymbol{O}$
2: **for** $i = n - 1$ downto 0 **do**
3: $\quad \boldsymbol{R_0} \leftarrow 2\boldsymbol{R_0}$
4: $\quad$ **if** $(k_i = 1)$ **then** $\boldsymbol{R_0} \leftarrow \boldsymbol{R_0} + \boldsymbol{P}$
5: **end for**
6: **Return** $\boldsymbol{R_0}$.

The method has the advantage that it provides a very efficient point multiplication but suffers from that it may leak information about the secret scalar k via physical side channels [KJJ99, MOP07]. In Simple Power Analysis (SPA) attacks, an adversary tries to recover the scalar k by measuring the power-consumption traces during scalar multiplication. If a difference between the operations of point addition and point doubling can be observed in the traces, then the scalar k is revealed bit-by-bit.

In [Cor99], Coron proposes a simple countermeasure that involves a dummy point addition operation if the scalar bit is set to 0. The so-called *double-and-add always* method actually prevents SPA attacks but becomes vulnerable to safe-error attacks, as shown in [YJ00]. A fault can be induced during the computation and an adversary can check whether the final result is correct or not. If the fault is injected during a dummy addition, the result is still correct and the corresponding bit of the scalar is 0. If the result is incorrect, the scalar bit is 1.

Another scalar-multiplication method that is commonly used is known as the *Montgomery ladder* [Mon87] and is depicted in Algorithm 2. The method presents several advantages for cryptographic applications.

First, the Montgomery ladder implicitly offers security against implementation attacks [JY03]. Since it performs the same curve operations in every loop iteration, an attacker cannot distinguish individual bits of the secret scalar by simply observing a side-channel trace and so prevents SPA-type attacks. Furthermore, the Montgomery ladder has a very regular structure and does not use dummy operations. This prevents fault-injection based safe-error attacks.

Second, group operations can be performed without the need of y-coordinates. Montgomery originally applied the technique to special (Montgomery form) el-

Algorithm 2 Montgomery ladder

Require: $\boldsymbol{P} \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}$
Ensure: $\boldsymbol{Q} = k\boldsymbol{P}$
1: $\boldsymbol{R_0} \leftarrow \boldsymbol{O}$; $\boldsymbol{R_1} \leftarrow \boldsymbol{P}$
2: **for** $i = n-1$ downto 0 **do**
3: $\quad b \leftarrow k_i$; $\boldsymbol{R_{1-b}} \leftarrow \boldsymbol{R_{1-b}} + \boldsymbol{R_b}$
4: $\quad \boldsymbol{R_b} \leftarrow 2\boldsymbol{R_b}$
5: **end for**
6: **Return** $\boldsymbol{R_0}$.

liptic curves as a way to speed up the elliptic curve factoring method. The technique was subsequently generalized to Weierstraß form curves [BJ02, FGKS02, IT02, IMT02].

Let $\boldsymbol{P_1} = (x_1, y_1)$ and $\boldsymbol{P_2} = (x_2, y_2)$ be two points on the elliptic curve $E : y^2 = x^3 + ax + b$ and x_D the x-coordinate of their difference $\boldsymbol{D} = \boldsymbol{P_2} - \boldsymbol{P_1}$. Then the x-coordinate of the sum $\boldsymbol{P_1} + \boldsymbol{P_2}$, say x_3, is given by

$$x_3 = \frac{2(x_1 + x_2)(x_1x_2 + a) + 4b}{(x_1 - x_2)^2} - x_D \ . \tag{5.4}$$

Alternatively, the x-coordinate of $\boldsymbol{P_1} + \boldsymbol{P_2}$ can be obtained in a multiplicative way as

$$x_3 = \frac{-4b(x_1 + x_2)(x_1x_2 - a)^2}{x_D(x_1 - x_2)^2} \ . \tag{5.4'}$$

The x-coordinate of $2\boldsymbol{P_2}$, say x_4, can be expressed from the x-coordinate of $\boldsymbol{P_2}$ as

$$x_4 = \frac{({x_2}^2 - a)^2 - 8bx_2}{4({x_2}^3 + ax_2 + b)} \ . \tag{5.5}$$

It is worth noticing that the Montgomery ladder keeps invariant the difference of the involved points throughout the entire scalar multiplication. Indeed, from the description in Algorithm 2, it is easily seen that $\boldsymbol{R_1} - \boldsymbol{R_0} = (\boldsymbol{R_1} + \boldsymbol{R_0}) - 2\boldsymbol{R_0}$ when $b = 0$, and $\boldsymbol{R_1} - \boldsymbol{R_0} = 2\boldsymbol{R_1} - (\boldsymbol{R_0} + \boldsymbol{R_1})$ when $b = 1$. Hence, $\boldsymbol{D} := \boldsymbol{R_1} - \boldsymbol{R_0} = \boldsymbol{P}$. Consequently, $\boldsymbol{R_1}$ will contain the value of $(k+1)\boldsymbol{P}$ at the end of the algorithm. When the calculation is performed using x-coordinates only, this allows one to recover the y-coordinate of $k\boldsymbol{P}$. Letting (x_1, y_1) the coordinates of $\boldsymbol{Q} = k\boldsymbol{P}$, (x_D, y_D) the coordinates of $\boldsymbol{P}$ and x_2 the x-coordinate of $(k+1)\boldsymbol{P}$, one has

$$y_1 = \frac{2b + (a + x_Dx_1)(x_D + x_1) - x_2(x_D - x_1)^2}{2y_D} \ . \tag{5.6}$$

This is useful for cryptographic schemes needing the y-coordinate of the resulting point; for example, in the verification of an ECDSA digital signature [Nat09].

5.3 New x-Coordinate Only Formulæ

This section presents new x-coordinate only formulæ for Weierstraß elliptic curves. We first provide the formulæ for addition and doubling of points in the co-Z coordinate representation. Second, we give formulæ for efficient differential addition-and-doubling in the same coordinate representation. Third, we discuss optimizations when applying dynamic ECC parameters and give appropriate formulæ to recover the full coordinates of the output point.

Let $\boldsymbol{P_1} = (X_1, Y_1, Z)$ and $\boldsymbol{P_2} = (X_2, Y_2, Z)$ be two points on the Weierstraß elliptic curve $E : Y^2Z = X^3 + aXZ^2 + bZ^3$ in *homogeneous*[1] projective coordinates that share the same Z-coordinate. Then, the x-coordinate of the addition of the two points, $\mathrm{x}(\boldsymbol{P_1} + \boldsymbol{P_2}) = (X_3, Z_3)$, can be evaluated as

$$\begin{cases} X_3 = 2(X_1 + X_2)(X_1X_2 + aZ^2) + 4bZ^3 - x_D Z(X_1 - X_2)^2 \\ Z_3 = Z(X_1 - X_2)^2 \end{cases}, \tag{5.7}$$

where $\boldsymbol{D} = \boldsymbol{P_2} - \boldsymbol{P_1} = (x_D, y_D)$ is the difference of the points $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ in affine coordinates. Note that the formula performs the point addition with x-coordinates only, thus no Y-coordinate is used. The point addition needs $5\mathsf{M} + 2\mathsf{S} + 1\mathsf{M}_a + 1\mathsf{M}_{4b}$ to get the resulting x-coordinate $\mathrm{x}(\boldsymbol{P_1} + \boldsymbol{P_2})$.

The x-coordinate of a point doubling operation, $\mathrm{x}(2\boldsymbol{P_2}) = (X_4, Z_4)$, needs $4\mathsf{M} + 3\mathsf{S} + 1\mathsf{M}_a + 1\mathsf{M}_{4b}$ and can be evaluated as

$$\begin{cases} X_4 = ({X_2}^2 - aZ^2)^2 - 8bZ^3X_2 \\ Z_4 = Z[4X_2({X_2}^2 + aZ^2) + 4bZ^3] \end{cases}. \tag{5.8}$$

Applying formulæ (5.7) and (5.8) to the Montgomery ladder needs three additional multiplications to project the resulting x-coordinates $\mathrm{x}(\boldsymbol{R_0}) = (X_3, Z_3)$ and $\mathrm{x}(\boldsymbol{R_1}) = (X_4, Z_4)$ to a common Z-coordinate. An equivalent representation for $\boldsymbol{R_0}$ and $\boldsymbol{R_1}$ can be obtained by evaluating

$$X_1' = X_3Z_4\,, \quad X_2' = X_4Z_3\,, \text{ and } \quad Z' = Z_3Z_4\,,$$

resulting in $\boldsymbol{R_0} \cong (X_1', Z')$ and $\boldsymbol{R_1} \cong (X_2', Z')$ sharing the same Z-coordinate. The total complexity for one Montgomery ladder loop iteration is therefore $12\mathsf{M} + 5\mathsf{S} + 2\mathsf{M}_a + 2\mathsf{M}_{4b}$. In the following, we show how to reduce the complexity for differential addition-and-doubling to only $\underline{9\mathsf{M} + 5\mathsf{S} + 1\mathsf{M}_a + 1\mathsf{M}_{4b}}$.

5.3.1 Differential Addition-and-Doubling

By combining the projective formulæ given by Eqs. (5.7) and (5.8) and class equivalences to have the same Z-coordinate, we obtain

[1] Previous works considered Jacobian coordinates when applying co-Z arithmetic on elliptic curves over fields of characteristic $\neq 2, 3$.

$$\begin{cases} X_1' = V[2(X_1 + X_2)(X_1X_2 + aZ^2) + 4bZ^3 - x_D ZU] \\ X_2' = U[({X_2}^2 - aZ^2)^2 - 8bZ^3X_2] \\ Z' = UVZ \end{cases}, \tag{5.9}$$

where $U = (X_1 - X_2)^2$ and $V = 4X_2({X_2}^2 + aZ^2) + 4bZ^3$. The points $\mathrm{x}(\boldsymbol{R_0}) = (X_1, Z)$ and $\mathrm{x}(\boldsymbol{R_1}) = (X_2, Z)$ get added and doubled resulting in the points $\mathrm{x}(\boldsymbol{R_0'}) = (X_1', Z')$ and $\mathrm{x}(\boldsymbol{R_1'}) = (X_2', Z')$. The formula reduces the complexity to $10\mathsf{M} + 4\mathsf{S} + 1\mathsf{M}_a + 1\mathsf{M}_{4b}$.

This can be further optimized by replacing the multiplication X_1X_2 involved in the previous formula with the equivalent expression $({X_1}^2 + {X_2}^2 - (X_1 - X_2)^2)/2$. The term can be multiplied with the leading factor 2 so that we finally obtain

$$\begin{cases} X_1' = V[(X_1 + X_2)({X_1}^2 + {X_2}^2 - U + 2aZ^2) + 4bZ^3 - x_D ZU] \\ X_2' = U[({X_2}^2 - aZ^2)^2 - 8bZ^3X_2] \\ Z' = UVZ \end{cases}. \tag{5.10}$$

Algorithm 3 Out-of-place differential addition-and-doubling in projective co-Z coordinate system using $9\mathsf{M} + 5\mathsf{S} + 14\mathsf{add} + 1\mathsf{M}_a + 1\mathsf{M}_{4b}$ and $8 + \{x_D, a, 4b\}$ registers.

Require: X_1, X_2, Z, x_D, a, $4b$
Ensure: X_1, X_2, Z

1: 1. $R_2 \leftarrow Z^2$
2. $R_3 \leftarrow a \times R_2$
3. $R_1 \leftarrow Z \times R_2$
4. $R_2 \leftarrow 4b \times R_1$
5. $R_1 \leftarrow {X_2}^2$
6. $R_5 \leftarrow R_1 - R_3$
7. $R_4 \leftarrow {R_5}^2$
8. $R_1 \leftarrow R_1 + R_3$
9. $R_5 \leftarrow X_2 \times R_1$
10. $R_5 \leftarrow R_5 + R_5$
11. $R_5 \leftarrow R_5 + R_5$
12. $R_5 \leftarrow R_5 + R_2$
13. $R_1 \leftarrow R_1 + R_3$
14. $R_3 \leftarrow {X_1}^2$
15. $R_1 \leftarrow R_1 + R_3$
16. $X_1 \leftarrow X_1 - X_2$
17. $X_2 \leftarrow X_2 + X_2$
18. $R_3 \leftarrow X_2 \times R_2$
19. $R_4 \leftarrow R_4 - R_3$
20. $R_3 \leftarrow {X_1}^2$
21. $R_1 \leftarrow R_1 - R_3$
22. $X_1 \leftarrow X_1 + X_2$
23. $X_2 \leftarrow X_1 \times R_1$
24. $X_2 \leftarrow X_2 + R_2$
25. $R_2 \leftarrow Z \times R_3$
26. $Z \leftarrow x_D \times R_2$
27. $X_2 \leftarrow X_2 - Z$
28. $X_1 \leftarrow R_5 \times X_2$
29. $X_2 \leftarrow R_3 \times R_4$
30. $Z \leftarrow R_2 \times R_5$

2: **Return** (X_1, X_2, Z)

This latter formula can be evaluated with $9\mathsf{M} + 5\mathsf{S} + 1\mathsf{M}_a + 1\mathsf{M}_{4b}$. Note that the formula overwrites the input coordinates X_1, X_2, and Z with the output variables X_1', X_2', and Z'. This avoids additional memory allocations for the output variables and avoids variable copying since the output variables serve as input variables for the next Montgomery loop iteration. Furthermore, the resulting points $\mathrm{x}(\boldsymbol{R_0}) = (X_1', Z')$ and $\mathrm{x}(\boldsymbol{R_1}) = (X_2', Z')$ share the same Z-coordinate and do not need any further updates. A detailed implementation is provided in Algorithm 3 which uses 8 intermediate registers. An implementation with 7 intermediate registers is given in Algorithm 4. Note that the elliptic-curve parameter b is always used in a quadruple representation so that it can be pre-computed and pre-stored as $4b$. In addition, all formulæ update the input variables with the resulting values using the same memory location. This avoids memory copies or pointer manipulations in hardware or software implementations. Finite field operations are denoted by $\times$ for multiplication, $\cdot^2$ for squaring, $+$ for addition, and $-$ for subtraction.

Algorithm 4 Out-of-place differential addition-and-doubling in projective co-Z coordinate system using $11\mathsf{M} + 4\mathsf{S} + 14\mathsf{add} + 1\mathsf{M}_a + 1\mathsf{M}_{4b}$ and $7 + \{x_D, a, 4b\}$ registers.

Require: X_1, X_2, Z, x_D, a, $4b$
Ensure: X_1, X_2, Z

1: 1. $R_1 \leftarrow X_1 \times X_2$
2. $R_3 \leftarrow Z^2$
3. $R_4 \leftarrow Z \times R_3$
4. $R_2 \leftarrow a \times R_3$
5. $R_1 \leftarrow R_1 + R_2$
6. $X_1 \leftarrow X_1 + X_2$
7. $R_3 \leftarrow X_1 \times R_1$
8. $X_1 \leftarrow X_1 - X_2$
9. $X_1 \leftarrow X_1 - X_2$
10. $R_1 \leftarrow 4b \times R_4$
11. $R_4 \leftarrow {X_1}^2$
12. $X_1 \leftarrow R_4 \times Z$
13. $R_3 \leftarrow R_3 + R_3$
14. $R_3 \leftarrow R_3 + R_1$
15. $Z \leftarrow X_2 \times R_4$
16. $R_4 \leftarrow R_1 \times X_2$
17. $R_1 \leftarrow {X_2}^2$
18. $R_2 \leftarrow R_1 + R_2$
19. $R_1 \leftarrow R_1 + R_1$
20. $X_2 \leftarrow x_D \times X_1$
21. $R_3 \leftarrow R_3 - X_2$
22. $X_2 \leftarrow R_1 \times R_2$
23. $X_2 \leftarrow X_2 + X_2$
24. $R_2 \leftarrow R_2 - R_1$
25. $R_1 \leftarrow R_4 + R_4$
26. $R_4 \leftarrow X_2 + R_4$
27. $X_2 \leftarrow {R_2}^2$
28. $R_1 \leftarrow X_2 - R_1$
29. $X_2 \leftarrow R_1 \times Z$
30. $Z \leftarrow X_1 \times R_4$
31. $X_1 \leftarrow R_3 \times R_4$

2: **Return** (X_1, X_2, Z)

5.3.2 (X, Y, Z) Recovery

We now give the formula for the recovery of the full projective coordinates for the output point $\boldsymbol{Q} = k\boldsymbol{P}$, from the x-coordinates $\boldsymbol{R_0} = (X_1, Z)$ and $\boldsymbol{R_1} = (X_2, Z)$ in co-Z representation available in memory at the end of the Montgomery ladder. First, we transform Eq. (5.6) from affine to projective coordinates and set $x_i = X_i/Z$ and $y_i = Y_i/Z$ ($i \in \{1, 2\}$). Then, we can calculate the representation of the output point $\boldsymbol{Q}$ in the projective coordinates $\boldsymbol{Q} \cong (X_1', Y_1', Z_1')$ with

$$\begin{cases} X_1' = DX_1A \\ Y_1' = 2[(CX_1 + aA)(C + X_1) - X_2(C - X_1)^2] + 4bB \\ Z_1' = DB \end{cases}, \tag{5.11}$$

where $A = Z^2$, $B = ZA$, $C = x_DZ$, $D = 4y_D$. X_1, X_2, and Z are the coordinates of the elliptic curve points after scalar multiplication and $\boldsymbol{D} = (x_D, y_D)$ represents the invariant of the Montgomery ladder in affine coordinates (namely, input point $\boldsymbol{P}$). The given formula needs $8\mathsf{M} + 2\mathsf{S} + 1\mathsf{M}_a + 1\mathsf{M}_{4b}$. The affine coordinates of the output point $\boldsymbol{Q}$ can then be calculated by one inversion and two multiplications, i.e., $\boldsymbol{Q} = (x_1, y_1) = (X_1' \cdot {Z_1'}^{-1}, Y_1' \cdot {Z_1'}^{-1})$. See Algorithm 5 for a detailed implementation.

Algorithm 5 Out-of-place (X, Y, Z)-recovery in projective co-Z coordinate system using $8\mathsf{M} + 2\mathsf{S} + 8\mathsf{add} + 1a + 1b4$ and $7 + \{x_D, y_D, a, 4b\}$ registers.

Require: X_1, X_2, Z, x_D, y_D, a, $4b$
Ensure: X_1, X_2, Z

1:
1. $R_1 \leftarrow x_D \times Z$
2. $R_2 \leftarrow X_1 - R_1$
3. $R_3 \leftarrow {R_2}^2$
4. $R_4 \leftarrow R_3 \times X_2$
5. $R_2 \leftarrow R_1 \times X_1$
6. $R_1 \leftarrow X_1 + R_1$
7. $R_3 \leftarrow a \times X_2$
8. $X_2 \leftarrow Z^2$
9. $R_2 \leftarrow R_2 + R_3$
10. $R_3 \leftarrow R_2 \times R_1$
12. $R_3 \leftarrow R_3 - R_4$
13. $R_3 \leftarrow R_3 + R_3$
14. $R_1 \leftarrow y_D + y_D$
15. $R_1 \leftarrow R_1 + R_1$
16. $R_2 \leftarrow R_1 \times X_1$
17. $X_1 \leftarrow R_2 \times X_2$
18. $R_2 \leftarrow X_2 \times Z$
19. $Z \leftarrow R_2 \times R_1$
20. $R_4 \leftarrow 4b \times R_2$
21. $X_2 \leftarrow R_4 + R_3$

2: **Return** (X_1, X_2, Z)

5.3.3 Optimizations for Dynamic ECC Parameters

If the curve parameters such as a, b are not fixed by the implementation and are chosen dynamically, the formula given in Eq. (5.10) can be optimized. In this case, the curve parameters have to be handled in RAM and their memory

Algorithm 6 Out-of-place differential addition-and-doubling in projective co-Z coordinate system using $10\mathsf{M} + 5\mathsf{S} + 13\mathsf{add}$ and 10 registers.

Require: X_1, X_2, $T_D = x_D Z$, $T_a = aZ^2$, $T_b = 4bZ^3$
Ensure: X_1, X_2, T_D, T_a, T_b

1: $R_2 \leftarrow X_1 - X_2$
2: $R_1 \leftarrow {R_2}^2$
3: $R_2 \leftarrow {X_2}^2$
4: $R_3 \leftarrow R_2 - T_a$
5: $R_4 \leftarrow R_3^2$
6: $R_5 \leftarrow X_2 + X_2$
7: $R_3 \leftarrow R_5 \times T_b$
8: $R_4 \leftarrow R_4 - R_3$
9: $R_5 \leftarrow R_5 + R_5$
10: $R_2 \leftarrow R_2 + T_a$
11: $R_3 \leftarrow R_5 \times R_2$
12: $R_3 \leftarrow R_3 + T_b$
13: $R_5 \leftarrow X_1 + X_2$
14: $R_2 \leftarrow R_2 + T_a$
15: $R_2 \leftarrow R_2 - R_1$
16: $X_2 \leftarrow {X_1}^2$
17: $R_2 \leftarrow R_2 + X_2$
18: $X_2 \leftarrow R_5 \times R_2$
19: $X_2 \leftarrow X_2 + T_b$
20: $X_1 \leftarrow R_3 \times X_2$
21: $X_2' \leftarrow R_1 \times R_4$
22: $R_2 \leftarrow R_1 \times R_3$
23: $R_3 \leftarrow R_2 \times T_b$
24: $R_4 \leftarrow {R_2}^2$
25: $R_1 \leftarrow T_D \times R_2$
26: $R_2 \leftarrow T_a \times R_4$
27: $T_b \leftarrow R_3 \times R_4$
28: $X_1 \leftarrow X_1 - R_1$
29: $T_D \leftarrow R_1$
30: $T_a \leftarrow R_2$
Return $(X_1, X_2, T_D, T_a, T_b)$

allocation can therefore be re-used as working space as soon as they are not needed. The following formulæ allows to save one register compared to the implementation of Eq. (5.10) with a and b permanently occupying a full register in RAM. By initializing three additional coordinates $T_a = aZ^2$, $T_b = 4bZ^3$, and $T_D = x_D Z$, we can evaluate

$$\begin{cases} T_D' = T_D W \\ T_a' = T_a W^2 \\ T_b' = T_b W^3 \\ X_1' = V[(X_1 + X_2)({X_1}^2 + {X_2}^2 - U + 2T_a) + T_b] - T_D' \\ X_2' = U[({X_2}^2 - T_a)^2 - 2X_2 T_b] \end{cases} \tag{5.12}$$

to perform a differential addition-and-doubling operation, where $U = (X_1 - X_2)^2$, $V = 4X_2({X_2}^2 + T_a) + T_b$, and $W = UV$. The given formula reduces the memory requirements by one working register and increases the performance by $1\mathsf{M}$ if the relation $\mathsf{M}_a = \mathsf{M}_b = 1\mathsf{M}$ is given (however, in practice, one has usually the relation $\mathsf{M}_a + \mathsf{M}_b = 1\mathsf{M}$; see § 5.5.2). Note that the formula does not involve either a, b, or x_D nor an explicit Z-coordinate throughout the scalar multiplication. See Algorithm 6 for a detailed implementation. The full coordinates

(X_1', Y_1', Z_1') can be recovered with $10\mathsf{M} + 3\mathsf{S}$ by evaluating

$$\begin{cases} X_1' = 4y_D x_D {T_D}^2 X_1 \\ Y_1' = {x_D}^3[T_b + 2(T_D X_1 + T_a)(X_1 + T_D) - 2X_2(X_1 - T_D)^2] \\ Z_1' = 4y_D {T_D}^3 \end{cases} . \quad (5.13)$$

5.4 In-Place vs. Out-of-Place Formulæ

Most descriptions of the elliptic-curve operations presented in the literature have claims of memory requirements and performances that assume that the finite-field operations can be performed *in-place*. That means that one source operand of the operation may be overwritten by the resulting value during the execution, e.g.

$$R_1 \leftarrow R_1 \circ R_2 \,,$$

where $R_1 \in \mathbb{F}_p$ and $R_2 \in \mathbb{F}_p$ are variables that store the source operands and R_1 is overwritten by the resulting value after execution of an operation $\circ$. In contrast, operations that do not overwrite the input operands are referred to as *out-of-place* operations, e.g.

$$R_3 \leftarrow R_1 \circ R_2 \,,$$

where $R_3 \in \mathbb{F}_p$ is an additional variable that stores the result of the operation.

In general, there exist several ways to implement modular operations in software and hardware. Most implementations use multi-precision arithmetic to process the large integer operands. That means that each operand is represented as a multiple-word data structure, i.e. $a = (a_{t-1}, \ldots, a_1, a_0)$ and $b = (b_{t-1}, \ldots, b_1, b_0)$, where t denotes the number of words. A 160-bit addition operation, for instance, that runs on a 16-bit processor, performs therefore ten additions by loading the input operands from memory, adding the two operands, and storing the result back to the memory. A subtraction is done in the same way, performing machine word subtractions instead of additions. However, during the computation both operations process each word of the operands sequentially and can thus perform the operation *in-place* at no cost in terms of memory or computational efficiency [HMV04].

In contrast, modular multiplication (and squaring) can be implemented in several ways. Basically, we can distinguish between *separated* and *integrated* modular multiplication [KAJ96, Koç95]. Separated modular multiplications perform the multiplication first and apply the reduction afterwards. In this approach, the result of the multiplication is stored in a temporary variable R_m which is then reduced in a separated step, e.g.

$$\begin{aligned} R_m &\leftarrow R_1 \times R_2, \\ R_1 &\leftarrow R_m \pmod{p}. \end{aligned}$$

This approach needs additional memory to store the temporary variable $R_m \in [0, 2^{2Wt})$, where W denotes the number of bits of a word (i.e. typically 8, 16, 32, or 64 bits).

The integrated (or interleaved) modular multiplication approach alternates between multiplication and reduction. There, partial products get reduced during the multiplication which avoids storing the double-sized result R_m and thus reduces the memory requirements significantly to the size of about the modulus $p \in [0, 2^{Wt})$ [Bla83, Slo85, Wol03, KAJ96, HWF08b, LSBV08]. However, for both multiplication types, the input operands cannot be overwritten with the resulting words because they are used not only once but multiple times throughout the algorithm. Therefore, implementations that allow *in-place* multiplications (and squarings) may need either an extra buffer to store the intermediate result $2^{Wt} \leq R_m < 2^{2Wt}$ or save the input operand to be overwritten during the computation. Formulæ for point operations in elliptic curves that involve *in-place* operations are thus very likely to require more memory in practice than claimed.

In this work, we propose *out-of-place* formulæ that use different source and destination variables to perform the modular multiplication and squaring operations. This guarantees that no additional memory is needed to perform the computation neither for software nor hardware implementations and that our formulæ will therefore meet our claims in all contexts.

5.5 Discussion

5.5.1 Security Analysis

The resistance to side-channel attacks and fault attacks is essential for the implementation of cryptography in embedded device. The given formulæ allow the use of traditional countermeasures against such attacks without disadvantages. As described in Section 5.2, the Montgomery ladder is well suited to the implementation of the scalar-multiplication method since it is resistant against SPA attacks [KJJ99, MOP07] as well as safe-error attacks [YJ00].

In addition, there exist several proposals to protect the Montgomery ladder against statistical attacks such as Differential Power Analysis (DPA) [KJJ99, MOP07]. One cheap but effective countermeasure against these attacks is the use of randomized projective coordinates as proposed by Coron [Cor99]. In our context, this countermeasure can be implemented by randomizing the intermediate points of the Montgomery ladder since they are represented in projective-coordinate representation as seen in Section 5.3. This can be done in Algorithm 2 at the cost of only two multiplications by randomizing the initial coordinates of the points $\boldsymbol{R_0}$ and $\boldsymbol{R_1}$ which are represented by the triplet $\{X_1, X_2, Z\}$ such that $\mathrm{x}(\boldsymbol{R_0}) = (X_1, Z)$ and $\mathrm{x}(\boldsymbol{R_1}) = (X_2, Z)$. Then, given a random value λ, the point $\mathrm{x}(\boldsymbol{P}) = (x_P, 1)$ is randomized to $\mathrm{x}(\boldsymbol{P'}) = (\lambda x_P, \lambda)$ for the initialization of $\mathrm{x}(\boldsymbol{R_0})$ and $\mathrm{x}(\boldsymbol{R_1})$ as follows:

$$\begin{cases} \mathrm{x}(\boldsymbol{R_1}) \leftarrow (\lambda x_P, \lambda) = \mathrm{x}(\boldsymbol{P'}) \\ \mathrm{x}(\boldsymbol{R_1}) \leftarrow \mathrm{doubling}(\boldsymbol{R_1}) = \mathrm{x}(2\boldsymbol{P'}) \\ \mathrm{x}(\boldsymbol{R_0}) \leftarrow (Zx_P, Z) = \mathrm{x}(\boldsymbol{P'}) \end{cases} . \tag{5.14}$$

This effectively randomizes every intermediate value during scalar multiplication and makes therefore DPA attacks ineffective. Note that the proposed Algorithms 3, 4, and 6 can be reused to avoid additional resources for the computation of the doubling operation. In this scenario, $(1, X_2, Z)$ is set as an input of the algorithm and only X_2 and Z of the output are further used as coordinates of the doubled point $2\boldsymbol{P'}$.

Third, to thwart fault injections during the scalar multiplication [Sko05, Sch09] a countermeasure that checks the resulting point can be applied. Checking that $x^3 + ax + b$ is a square may seem conceivable, unfortunately that may not detect if the point belongs to the twist curve instead of the original curve and would leave the implementation vulnerable to attacks such as the one introduced by Fouque et al. [FLRV08]. Another check consists in verifying that the coordinates of the resulting point satisfy the curve equation, in which case the recovery of the y-coordinate is required. However that can be done in an efficient way with projective coordinates i.e. $Z(Y^2 - bZ^2) = X(X^2 + aZ^2)$ [EL09]. This countermeasure effectively protects against fault attacks on the Montgomery ladder even when implemented with x-coordinate only formulæ [FLRV08].

5.5.2 Performance Analysis

We now compare our formulæ with existing differential addition-and-doubling formulæ. Comparing one formula with another is not straightforward because the complexity ratio of the field arithmetic operations involved may vary according to the underlying implementation as well as the usage context. Hence we provide the global complexity of each algorithm along with figures corresponding to some assumptions that are made based on experience and previous works.

Thus, we first adopt the common assumption that the squaring operation is faster than a multiplication with the weighting $1\mathsf{S} = 0.8\mathsf{M}$ [HMV04]. The cost of additions and subtractions are usually neglected when evaluating the complexity of the formulæ. However, according to several previous works [LH04, MvOV97, GV10] it may be relevant to take these operations into account since in practice they have reported ratios from $1\mathsf{add} = 0.1\mathsf{M}$ up to $1\mathsf{add} = 0.3\mathsf{M}$. Hence, in the following we will consider both cases, by first considering additions negligible and then the worst case where $1\mathsf{add} = 0.3\mathsf{M}$. Besides, a special case can also be made for the multiplications involving the curve parameters and especially the parameter a because several standardized curves have a set to -3. In any case, this assumption can be applied without loss of generality because a curve isomorphism can be used to reduce a to a small relative integer [IEE00, §§ A.9.5 and A.10.4] (see also [BJ03]). Subsequently, we will assume that a multiplication with a takes 2 additions, i.e. $1\mathsf{M}_a = 2\mathsf{add}$. On the other hand, the multiplication

Table 5.1: Complexity of scalar multiplications per bit of scalar.

Method	Costs	M/bit[a]	M/bit[b]
Algorithm 6	**10M + 5S + 13add**	**14**	**17.9**
Algorithm 3	**9M + 5S + 1M$_a$ + 1M$_{4b}$ + 14add**	**14**	**18.8**
Izu et al. [IMT02]	10M + 4S + 2M$_a$ + 1M$_b$ + 18add	14.2	20.8
Goundar et al. [GJM10]	8M + 7S + 3M$_a$ + 1M$_b$ + 18add	14.6	21.8
Algorithm 4	**11M + 4S + 1M$_a$ + 1M$_{4b}$ + 14add**	**15.2**	**20.0**
Fischer et al. [FGKS02]	10M + 5S + 2M$_a$ + 2M$_b$ + 14add	16	21.4
Algorithm 7	**15M + 6S + 2M$_{4b}$ + 25add**	**21.8**	**29.3**

[a] M$_b$ = 1M ; S = 0.8M ; 1M$_a$ ≃ 0 ; 1add ≃ 0 (negligible)
[b] M$_b$ = 1M ; S = 0.8M ; 1M$_a$ = 2add ; 1add = 0.3M

with b (or any fixed pre-computed multiple e.g. $4b$ denoted M$_{4b}$) is considered as a regular modular multiplication of cost M$_b$ = 1M.

In the following, we compare different low-memory scalar multiplication formulæ first sorted by performance in Table 5.1 and then sorted by memory requirements in Table 5.2.

Table 5.1 shows the efficiency of the formulæ we proposed in Section 5.3. For a comparison, we also added a new formula given in Algorithm 7 that use only 6 working registers but requires *in-place* operations and a fixed curve parameter $a = -3$.

It shows that Algorithm 6 and Algorithm 3 are more efficient than any previous works found in literature with provided that 1S ≥ 0.5M and 1M$_a$+1M$_b$ ≥ 1M. The performance improvement is significant (up to 14 % less multiplications per bit) when adopting the usual assumption that 1S ≥ 0.8M and 1M$_b$ = 1M. For memory-constrained devices where the curve parameter a and b are fixed, Algorithm 3 is shown to be more efficient than previous works as long as 1S ≥ 0.5M which is generally admitted. One can also remark that the benefits of our approach increases when the squaring is performed using the multiplication instead of a dedicated implementation (for program or hardware saving) as well as when the secondary operations such as additions and subtractions are not negligible

Table 5.2: Memory requirements of scalar multiplications.

Method	Working registers	In-place[a] memory	Constants	Total
Algorithm 7	**6 reg.**	+1 reg.	**{$x_D, a, 4b$}**	**10 reg.**
Algorithm 4	**7 reg.**	-	**{$x_D, a, 4b$}**	**10 reg.**
Izu et al. [IMT02]	7 reg.	+1 reg.	{x_D, a, b}	11 reg.
Goundar et al. [GJM10]	7 reg.	+1 reg.	{x_D, a, b}	11 reg.
Algorithm 3	**8 reg.**	-	**{$x_D, a, 4b$}**	**11 reg.**
Fischer et al. [FGKS02]	8 reg.	+1 reg.	{$x_D, a, 4b$}	12 reg.
Algorithm 6	**10 reg.**	-	-	**10 reg.**

[a] In-place operations require additional memory to perform multiple-precision arithmetic operations (see Section 5.4).

(as observed in practice). Algorithm 7 provides worst performance compared to all other given algorithms.

Table 5.2 lists the memory requirements of the scalar multiplication methods. For constrained devices where the elliptic-curve parameters x_D, a, b or $4b$ are hard-coded or stored in read-only memory, Algorithm 7 and Algorithm 4 provide the lowest memory requirements. Algorithm 7 requires only 6 working registers but needs additional memory to perform the ECC operations *in-place* (see Section 5.4 for a detailed description). In contrast, Algorithm 4 needs the same amount of working registers but performs all operations *out-of-place*. The memory requirements are the same but it shows that the *out-of-place* formula of Algorithm 4 is much faster than the *in-place* Algorithm 7 which emphasizes the use of *out-of-place* formulæ in that context.

In a scenario where the curve parameters cannot be set during the design time of the device or if they can not be processed directly from the read-only memory as it is in the case with most cryptographic accelerators, Algorithm 6 becomes equivalent in terms of memory requirement to Algorithm 7 and 4 while being faster as shown in Table 5.1.

5.6 Summary and Conclusion

In this chapter, we presented new formulæ for fast and memory-efficient scalar multiplication on elliptic curves over prime fields. The proposed formulæ use *out-of-place* operations, namely the source and destination variables of finite-field multiplications are always different. This guarantees that neither additional memory is needed nor additional operations have to be executed to perform multiple-precision arithmetic operations in both software or hardware implementations. Furthermore, the given formulæ outperform existing solutions by using a co-Z coordinate representation. The formulæ can be applied on general elliptic curves and allow the integration of conventional countermeasures against implementation attacks. They can be efficiently applied in low-resource implementations of RFIDs, smart cards, or other embedded systems.

Algorithm 7 In-place differential addition-and-doubling in projective co-Z coordinate system using $17\mathsf{M} + 6\mathsf{S} + 25\mathsf{add}$ and 6 registers ($a = -3$).

Require: X_1, X_2, Z, x_D, $4b$, $a = -3$
Ensure: X_1, X_2, Z

1. $R_1 \leftarrow X_1 \times X_2$
2. $R_2 \leftarrow Z^2$
3. $R_3 \leftarrow R_2 \times Z$
4. $R_1 \leftarrow R_1 - R_2$
5. $R_1 \leftarrow R_1 - R_2$
6. $R_1 \leftarrow R_1 - R_2$
7. $R_2 \leftarrow 4b \times R_3$
8. $R_3 \leftarrow X_1 - X_2$
9. $X_1 \leftarrow X_1 + X_2$
10. $R_1 \leftarrow X_1 \times R_1$
11. $R_1 \leftarrow R_1 + R_1$
12. $R_1 \leftarrow R_1 + R_2$
13. $X_1 \leftarrow R_3^2$
14. $X_1 \leftarrow Z \times X_1$
15. $X_1 \leftarrow x_D \times X_1$
16. $R_1 \leftarrow R_1 - X_1$
17. $R_1 \leftarrow x_D \times R_1$
18. $R_3 \leftarrow X_2 + Z$
19. $R_2 \leftarrow X_2 \times Z$
20. $X_2 \leftarrow X_2 - Z$
21. $X_2 \leftarrow X_2 - Z$
22. $X_2 \leftarrow X_2 - Z$
23. $R_3 \leftarrow R_3 \times X_2$
24. $X_2 \leftarrow X_2 + Z$
25. $X_2 \leftarrow X_2 + Z$
16. $X_2 \leftarrow X_2 + Z$
17. $R_2 \leftarrow R_2 + R_2$
18. $R_3 \leftarrow R_3 + R_2$
19. $R_3 \leftarrow R_3 \times X_2$
20. $R_3 \leftarrow R_3 + R_3$
21. $R_3 \leftarrow R_3 + R_3$
22. $R_2 \leftarrow Z^2$
23. $R_2 \leftarrow R_2 \times Z$
24. $R_2 \leftarrow 4b \times R_2$
25. $R_3 \leftarrow R_3 + R_2$
26. $R_3 \leftarrow R_3 \times Z$
27. $R_1 \leftarrow R_1 \times R_3$
28. $R_3 \leftarrow R_3 \times X_1$
29. $R_2 \leftarrow R_2 \times X_2$
30. $R_2 \leftarrow R_2 + R_2$
31. $X_2 \leftarrow X_2^2$
32. $Z \leftarrow Z^2$
33. $X_2 \leftarrow X_2 + Z$
34. $X_2 \leftarrow X_2 + Z$
35. $X_2 \leftarrow X_2 + Z$
36. $X_2 \leftarrow X_2^2$
37. $X_2 \leftarrow X_2 - R_2$
38. $X_2 \leftarrow X_2 \times X_1$
39. $X_1 \leftarrow R_1$
40. $Z \leftarrow R_3$

Return (X_1, X_2, Z)

6

A Low-Resource Hardware Processor for RFID Devices Supporting AES and ECDSA

The vast majority of security failures occur at the level of implementation detail.
— Ross Anderson

In this chapter, we will apply the results obtained in Chapter 5 to design a very efficient and compact hardware implementation of a cryptographic processor. We make also use of our results discussed in Chapter 2, Chapter 3, and Chapter 4 to integrate countermeasures against side-channel and fault attacks. The proposed processor is able to perform both signing of data using ECDSA based on the NIST elliptic curve over $\mathbb{F}_{p192}$ and AES (encryption and decryption) with 128-bit keys. Our implementation targets low-resource devices such as passive RFID tags. This focus faces fierce requirements concerning chip area (costs) and power consumption (due to the contactless operation). We meet the ambitious design goals by using a sophisticated hardware architecture where the main components memory, datapath, and controlling engine are heavily reused by all implemented algorithms. Using a 16-bit datapath with a multiply-accumulate unit and a dedicated RAM macro instead of a flip-flop based memory minimizes the chip area. In order to accomplish the demanding controlling effort, a hierarchical approach using a microcode control has been implemented which is steered using a highly optimized integrated 8-bit microcontroller. Our results provide an optimum between area and speed in comparison to existing elliptic-curve processors over $\mathbb{F}_{p192}$ and additionally provide AES encryption and decryption as

well as higher-layer functions to generate digital signatures using ECDSA. The proposed processor has a chip area of 21 674 GEs and signs a digital message within 863 109 clock cycles.

This chapter includes outcomes from the Austrian government funded project *CRYPTA*. It has been a joint work together with Martin Feldhofer, Thomas Plos, and Johannes Wolkerstorfer. Results have been published in the Workshop on RFID Security (RFIDSec 2010) [HFP10], the Workshop on RFID/USN Security and Cryptography (RISC 2010) [FAH+10], and the Workshop in Information Security Theory and Practice (WISTP 2011) [HFW11].

First, we will give a brief overview about related work and the state-of-the-art in designing low-resource implementations of AES and ECC in Section 6.1. Afterwards, we will describe the overall system and hardware architecture of our proposed processor in Section 6.2. Next, we will describe the arithmetic-level implementation in Section 6.3 where we will focus on the implemented algorithms to perform ECC modular arithmetics. In Section 6.4, we will describe the implemented algorithms which are SHA-1, AES, and ECDSA. Section 6.5 will cover the system-level implementation that contains the random-number generation and integration of the implemented algorithms in cryptographic protocols. The results of our work are given in Section 6.6 and we will conclude in Section 6.7.

6.1 Related Work

In the following, we will describe related work for low-resource AES implementations in hardware. Afterwards, we will give related work on ECC-based hardware implementations.

6.1.1 Low-Resource AES Hardware Designs

One landmark paper that reports a low-resource implementation of AES is due to M. Feldhofer et al. [FDW04] in 2004. They have been the first who presented a low-resource AES implementation and demonstrated the feasibility of AES for passive RFID tags. In particular, they proposed a protocol that uses AES to allow entity authentication of tags and readers. Their implementation needs 3 595 GEs and performs a 128-bit encryption within 1 032 clock cycles. The proposed design consumes around 8 μA of mean current at 100 kHz operating frequency. The chip has been fabricated on a 350 nm CMOS process technology and has 3.3 volts supply voltage.

P. Hämäläinen et al. [HAHH06] presented another AES architecture in 2006. Their (encryption-only) design has been synthesized using a 130 nm CMOS process technology. It needs only 3 100 GEs and consumes 3.1 μA of mean current at 100 kHz and 1.2 volts. A 128-bit AES encryption needs 152 clock cycles. In the same year, J.-P. Kaps et al. [KS06] and M. Kim et al. [KRCJ06] proposed a similar solution that need around 4 000 GEs using a 130 nm and 250 nm target library. Both designs consume around 2 μA of mean current at 100 kHz and 1.2

volts (Kaps) and 2.5 volts (Kim). The design of Kaps needs 534 clock cycles and the design of Kim needs 870 clock cycles.

In 2008, M. Feldhofer et al. [FP08] proposed an encryption and decryption AES hardware module suitable for RFID tags that needs 3 648 GEs. The chip (called *TINA*) consumes 2.9 μA of mean current at 1.5 volts and performs a 128-bit encryption in 1 044 clock cycles. Note that the authors also proposed a side-channel resistant implementation of TINA (called *secureTINA*). They integrated several countermeasures including the Masked Dual-Rail Precharge Logic (MDPL) style as a masking countermeasure, dummy operations, and shuffling of AES *State* bytes. *SecureTINA* needs 19 500 GEs and consumes 13.9 μA of mean current at 100 kHz and 1.5 volts. Without dummy operations and only shuffling enabled, it needs 1 076 clock cycles to perform an encryption. For 15 dummy operations, encryption needs 1 736 cycles. Note that about 80 % of the area has been needed only for the masking countermeasure using MDPL.

E. Trichina et al. [TKL05] proposed another side-channel resistant implementation of AES in 2004. Their design needs 16 390 GEs and performs an AES encryption within 160 clock cycles. The chip consumes around 18 μA of mean current at 100 kHz and 1.8 volts using a 180 nm CMOS process technology.

6.1.2 Low-Resource ECC Hardware Designs

There exist many proposals for low-resource hardware implementations of ECC processors and ECC co-processors. Many of these implementations are based on reconfigurable hardware platforms like Field-Programmable Gate Arrays (FPGA). Examples for implementations of elliptic curve (EC) co-processors over $\mathbb{F}_{2^m}$ are the work of G. Orlando and C. Paar [OP00], S. Okada et al. [OTIT00], H. Eberle et al. [EGSG03], and X. Guo [GFSV09]. Examples of FPGA implementations of EC processors over $\mathbb{F}_p$ are due to G. Orlando and C. Paar [OP01], N. Gura et al. [GES02], C. McIvor et al. [MMM04], and T. Güneysu et al. [GP08].

There also exist many standard-cell based solutions of ECC such as proposed by E. Öztürk et al. [ÖSS04], S. Kumar et al. [KP06], and B. Örs et al. [OBP02]. However, the majority of the publications focus on the implementation of efficient point-multiplication datapaths and do not provide necessary operations for higher-layer protocols. Only a few publications describe ECC processors that involve additional circuit logic to be integrated as a stand-alone module. A. Satoh et al. [ST03] proposed a dual-field processor supporting high-level functions by using a dedicated sequencer unit. Next to the basic modular-arithmetic unit, they support prime number generation, CRT-RSA, DSA, and ECDSA but reported only hardware results for the point-multiplication unit.

L. Batina et al. [BGK+06] presented a low-resource ECC co-processor over binary fields ($\mathbb{F}_{2^{131}}$, $\mathbb{F}_{2^{139}}$, and $\mathbb{F}_{2^{163}}$) in 2006. The co-processor implements the Okamoto identification protocol [Oka93] and the Schnorr identification protocol [Sch90] in hardware (see also Chapter 4 for a description of the protocols). The chip was especially designed for RFID tags and needs between 8 580 and 11 000 GEs of area (without involving RAM memory) dependent on the digit size and chosen field type.

Y. K. Lee et al. [LSBV08] proposed a co-processor that implements ECC over binary fields $\mathbb{F}_{2^{163}}$ in 2008. It provides entity authentication services using identification protocols like the one of Schnorr [Sch90]. They integrated a tiny microcontroller for higher-level arithmetics which facilitates the implementation and adaption of different RFID authentication protocols. In the same year, another application-near solution was presented by D. Hein et al. [HWF08b]. They described an ECC co-processor over $\mathbb{F}_{2^{163}}$ that was fabricated on a 180 nm CMOS process. The chip was designed for RFID applications and integrates a digital RFID front-end supporting the ISO/IEC 18000-3 [Int04b] standard. Nevertheless, they also omitted operations in $\mathbb{F}_p$ that are needed for ECDSA. H. Bock et al. [BBD+08] published similar results of an $\mathbb{F}_{2^{163}}$ co-processor suitable for RFID applications. Next to an RNG, the chip performs a Diffie-Hellman based authentication protocol, an RFID-protocol interface according to ISO/IEC 15693 [Int01b], and integrates also several countermeasures against implementation attacks.

ECC processors over $\mathbb{F}_p$ have been proposed by J. Wolkerstorfer [Wol05] and F. Fürbass et al. [FW07] in 2005 and 2007. They described an ECC processor over the recommended NIST $\mathbb{F}_{p192}$ elliptic curve [Nat00]. The processor mainly targets ECDSA signature generation. They reported results for point multiplication but they neither included a Pseudo-Random Number Generator (PRNG) nor the hashing of messages to complete the signing process.

The first ECDSA processor that supports ECDSA over $\mathbb{F}_{p192}$ has been reported by A. Auer [Aue08] in 2008. He implemented a finite-state machine based processor that signs a message in 1 031 000 clock cycles. The design has been synthesized using a 350 nm CMOS process technology and has an area of about 25 000 GEs including a standard-cell based RAM implementation.

In 2010, we first presented a processor that supports ECDSA using a microcontroller, a microcode-control, and a dual-port RAM macro [HFP10]. Our controller signs a digital message within 859 188 clock cycles and needs an area of 19 115 GEs. The architecture of this chip is part of this chapter and described in the next sections. Note that we improved these results and included also AES to the design as described later.

T. Kern et al. [KF10] presented a similar processor supporting ECDSA using the SECG elliptic curve over $\mathbb{F}_{p160}$ [Cer00b, Cer00a]. Their implementation needs 18 247 GEs and 511 864 cycles to sign a message.

Another ECDSA processor has been reported by E. Wenger et al. [WFF11] in 2011. They implemented ECDSA using the NIST curve over $\mathbb{F}_{p192}$. In contrast to others, they used a single-port RAM macro and implemented a 16-bit microcontroller to efficiently handle the ECC operations. Their processor needs 11 686 GEs and requires 1 377 000 cycles to sign a message.

6.2 System Overview

In this section, we will describe the general requirements of our system in terms of area, power, speed, and security. Afterwards, we will give an overview of the

implemented system and the architecture of our hardware design.

6.2.1 Objectives and Requirements

The main objectives and requirements of a low-resource processor are low chip area, low power consumption, appropriate speed, and a sufficient level of security:

Chip Area From an economical point of view, chip area is one of the most important issues for hardware designs. This is because the lower the chip area, the lower will be the production costs in general. In fact, small chip designs can be fabricated more cost efficiently than larger chip designs which is due to functional defects and the fabrication yield of a production [Kae08]. The size of a chip plays therefore a major role in the total costs of a product on the market.

Low Power Achieving a low-power design is a technical requirement that is important in the field of passively or battery-powered applications. There, it is necessary that the chip consumes as less power as possible. A typical example of such an application are contactless smart cards or RFID tags, which get powered by the field of an external reader. In such an environment, hardware designs have to work with low-power conditions and should operate even at a larger distance to the reader.

Speed Another requirement is the time that a processor requires to perform a cryptographic operation. If a processor needs several seconds to provide a certain cryptographic service then it would be useless in various applications, for instance, contactless payment systems or access control. The faster the operation the broader will be the application range where the processor can be applied. It is therefore necessary to minimize the required number of clock cycles of a processor to be applicable in practice.

Security Last but not least, cryptographic processors are susceptible to implementation attacks as described in the previous chapters of this thesis. A requirement is therefore to provide a sufficient level of protection against these attacks to avoid extraction of secret information. During our investigations it has been shown that it is advisable to implement several proposed countermeasures and to combine them as a protection against attacks [MOP07].

For our processor, we decided to implement both ECDSA and AES in hardware. As already described in Chapter 4, ECDSA provides several cryptographic services such as data integrity, non-repudiation, and authentication. In particular, dependent on the used protocol, ECDSA provides entity as well as message authentication that can be used in many open-loop RFID applications. The public-key primitive of ECC generally avoids key-distribution problems and complies with already existing infrastructures such as the X.509 Public-Key Infrastructure (PKI) [Int08b]. This infrastructure allows authentication of devices

and products also by offline entities in the field which is an important feature in future technologies such as in the Internet of Things.

The support of AES in our processor is advantageous due to three reasons. First, the processor is able to provide much faster authentication using the symmetric cryptographic primitive. This is necessary for most applications and cannot be achieved with our ECDSA implementation. Second, the costs for including AES is rather low since we can reuse RAM, the accumulator register, and the controller in our hardware design. It shows that this is cheaper in terms of chip area instead of combining implementations in dedicated hardware modules. Third, a combined ECDSA, SHA-1, and AES unit is appealing to a larger market because cryptographic facilities get close to smart cards, which are typically more expensive.

6.2.2 Hardware Architecture

We decided to implement a 16-bit hardware architecture. It has shown that a 16-bit word size provides an optimum to reduce the chip area and the power consumption while keeping the required number of clock cycles within limitations. Larger word sizes need larger datapath components such as multipliers, adders, multiplexers, and registers. This increases both the chip area and the power consumption of the device but makes the execution faster. A smaller word size, in contrast, reduces the size of the chip and reduces the power consumption significantly. The computational performance is lower and limits the use of the processor for many applications. Thus, we decided to use a 16-bit architecture which provides a good trade-off between area, power, and speed.

For ECDSA, we implemented all operations over prime fields and made use of the smallest recommended NIST Weierstrass elliptic curve which is over $\mathbb{F}_{p192}$. There are mainly two reasons for that choice. First, our processor requires modular integer operations to perform the final signing process in ECDSA and thus encourages the use of prime-field arithmetics. A binary extension-field processor would benefit in a better performance gain but needs an increased overhead for the required integer multiplier. An ECC processor over $\mathbb{F}_p$ already provides all mandatory operations without the need of additional logic circuits which further emphasizes our decision to meet the requirements of a low-area design. Second, fixing the implementation to a standardized elliptic curve provides interoperability with existing applications like X.509 public-key infrastructures. A recommended elliptic curve can further allow several optimizations in hardware like the NIST modular reduction to gain additional performance.

AES has been implemented in addition to ECDSA. As stated before, it reuses several components such as the memory, accumulator register, and controller. The overhead for AES is therefore very low because ECDSA dominates the memory requirements and the controlling effort.

The processor can be accessed by a memory-mapped I/O. Via an Advanced Microcontroller Bus Architecture (AMBA) [ARM97] interface it is possible to write and read data to and from the memory. The bus allows writing and reading from RAM, EEPROM, and the instruction register of the controller.

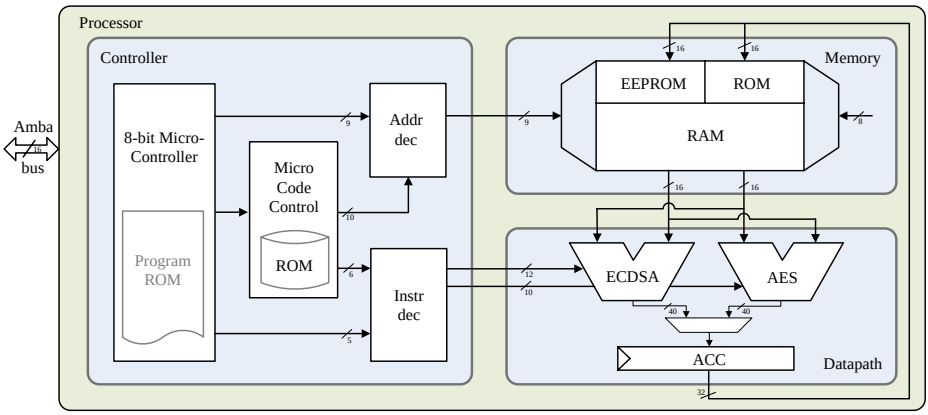


Figure 6.1: Architecture of the Cryptographic Processor.

The ROM can only be accessed by the controller during signing of data or encryption and decryption. The AMBA interface consumes nearly no power when no communication takes place and provides a standardized interface to our processor.

The cryptographic processor consists of three main components as depicted in Figure 6.1. The first component is the memory which holds data during computation, constants like curve parameters, and also non-volatile data like the private key for ECDSA and the secret key for AES. Note that most memory is needed for ECC operations. AES as well as SHA-1 can reuse the existing memory which reduces the additional overhead for those algorithms. The second component is the datapath. It performs the arithmetic and logic operations for ECDSA (including SHA-1) and AES in separated paths. For all implemented algorithms, we minimized the required hardware resources by reusing components like the memory and the controller to meet the low-resource requirements. The third module is the controller, which is responsible for sequencing the desired algorithms and the generation of the control signals for the memory and the datapath unit. In very complex algorithms and protocols like ECDSA with implicit random-number generation, the controlling effort in terms of design complexity and chip area gets more and more dominant. Hence, we investigated a totally new concept where a microcode-control approach makes the implementation more flexible but keeps also the hardware complexity low compared to dedicated finite-state machines.

6.2.3 Memory Unit

The memory unit comprises the three types RAM, ROM, and EEPROM in a 16-bit linear addressable dual-port memory space. The 128×16-bit dual-port RAM is realized as a dedicated macro block. This halves the chip area of this memory resource. In detail, ECDSA needs 7×192 bits for calculating the point

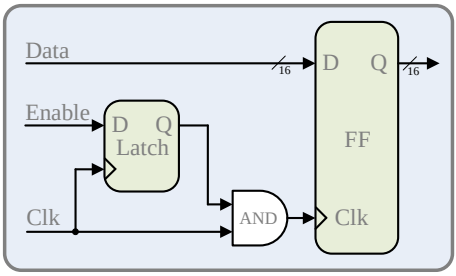


Figure 6.2: A clock-gating cell as clock input for a register.

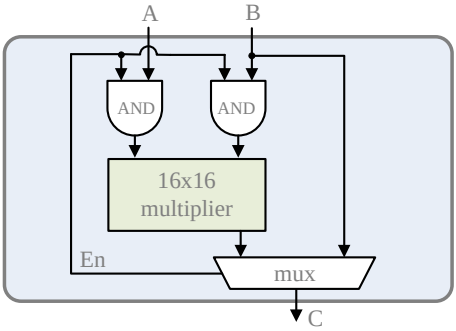


Figure 6.3: Operands A and B get isolated from a 16×16 multiplier if the enable signal is low.

multiplication, one 192-bit value to store the message that has to be signed, and one 192-bit value for the ephemeral key k. Additionally, we reserved 192 bits for storing the seed that is used in both ECDSA and AES to generate the needed random numbers. The ROM circuit stores 128 16-bit constants like ECC parameters, SHA-1, and AES constants and is implemented as an unstructured mass of standard cells. The EEPROM stores non-volatile data like the ECDSA private key, the public-key certificate, the AES secret key, and potentially other user-specific data up to 4K bits, which can be written in a personalization phase or during the protocol execution. Most of the related work does not account for this important real-world requirement.

Furthermore, we included clock-gating cells to each register. Clock gating is a technique that is commonly used for low-power hardware designs. Sequential elements like registers and flip-flops need a considerable amount of power when they are active. Thus, it is recommended to disable specific clock nets that are not used during certain computations. For this, a clock-gating cell has been inserted into every clock input of a synchronous register. Figure 6.2 shows such a clock-gating domain cell. The cell is mainly composed of a latch (which prevents glitches in the clock signal) and an AND gate. If the cell is enabled, the clock signal is passed through to the register otherwise it is blocked. Thus, unnecessary toggling of flip-flops is prevented which reduces the power consumption of the memory unit significantly. Most synthesis tools such as the Cadence RTL compiler [Cad] already provide such techniques out-of-the-box so that only a parameter has to be set to enable that feature. The synthesis tool simply replaces the original design (usually a multiplexer with enable signal) and automatically applies clock-gating cells instead.

6.2.4 Datapath Unit

The datapath of our processor is shown in Figure 6.4. It is mainly composed of ECDSA and AES components. Both ECDSA and AES share one single 40-bit accumulator register which pursues the strategy of reusing components to minimize the chip area. The 40-bit register is used as an accumulator for the implemented multiply-accumulate architecture in ECDSA (see Section 6.3 for more details) as well as intermediate storage for AES. The reason why we clearly separated the datapaths is due to the ability to enable or disable one of the components independently. First, generic elements make the processor more flexible and scalable for a broad range of applications. It allows the designer to personalize the processor which makes it more comfortable in terms of reuse and modifications. Second, clear separations of chip components reduce the power consumption of the chip by using an *operand-isolation* technique (also often referred as to *sleep logic*) to our datapath architecture.

Operand isolation is a technique that uses an enable signal to turn on or off the activity in a datapath architecture. Figure 6.3 shows the operand-isolation flow. If the enable signal is high, the operands are used by the datapath and the result is stored in the register. If it is low, the operands get isolated from the circuit and no activity occurs in that clock cycle. Thus, if AES is enabled in our datapath architecture, the entire ECDSA datapath gets isolated. This reduces the power consumption significantly since the most power is consumed by the multiplier unit. It is therefore recommended to apply such techniques to reduce the total power dissipation of the chip. Like clock gating, common synthesizers already include such techniques that replace the original elements with operand-isolation elements.

The AES Datapath

The AES datapath is mainly composed of an S-box submodule, a MixColumns submodule, five multiplexers, one XOR gate, and two 16-to-8 bit converters. While the entire architecture uses 16 bits, we decided to use only 8 bits of the datapath to perform AES operations. First, all AES operations can be efficiently implemented with 8 bits only. This reduces the needed amount of gates in the datapath and allows to temporarily store one column (four bytes of the *State*) in the common 40-bit accumulator. Second, we reused the remaining 8 bits of the datapath to implement side-channel countermeasures which are described in Section 6.3.

Most of the AES datapath is needed for the S-box and MixColumns operation. For the S-box operation, we calculated the substitution values using combinatorial logic instead of a look-up table. The MixColumns operation is also realized as an individual submodule which transforms one byte of the AES *State* using one single clock cycle. This architecture has been taken from the low-power AES implementation of M. Feldhofer et al. [FDW04].

The ShiftRows and AddRoundKey operations, in contrast, can be realized without expensive logic circuits. These operations need several controlling sig-

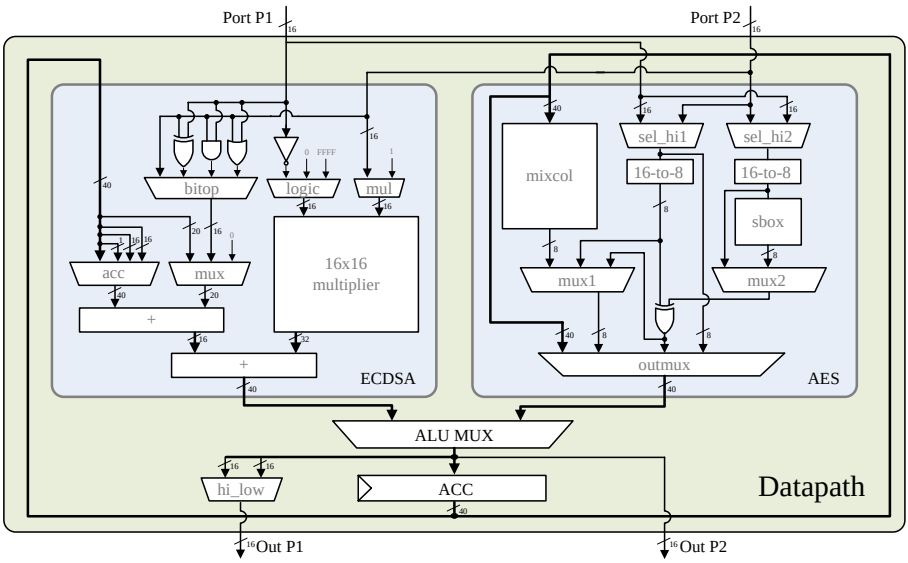


Figure 6.4: Architecture of the ECDSA and AES Datapath.

nals and one XOR gate. The AES constant *Rcon* has been externally stored in the ROM memory. Furthermore, as already stated before, we integrated an *operand-isolation* technique to reduce the power consumption of the processor. If AES is disabled, the operands for the AES operations get isolated from the ECDSA datapath. This eliminates unnecessary power dissipation and reduces the power consumption of the processor by about 13 %.

The ECDSA Datapath

The ECDSA datapath is mainly composed of a 16×16-bit multiplier, two 40-bit adders, and five multiplexers. For low-area reasons, we decided to use a 16-bit multiply-accumulate (MAC) architecture to perform a finite-field multi-precision multiplication. For this, partial products are calculated and accumulated in the common register to perform a multiplication in the product-scanning (Comba) method [HMV04] (see Section 6.3 for a detailed description of the implemented algorithms). Compared to all other parts of the datapath, the multiplier needs the most circuit area and consumes the most power. It also plays a major role in the path delay of the processor and the maximum frequency limit.

The SHA-1 Datapath

ECDSA includes the SHA-1 hash algorithm to hash the message before signing. Thus, we decided to integrate all needed components to perform SHA-1 operations into the ECDSA datapath. These are four additional 16-bit logic gates, i.e. AND, OR, XOR, and NOT which are very cheap in terms of chip area. Fig-

ure 6.5 shows the gates of the SHA-1 datapath. The logic operations are directly connected to port P1 and port P2 of the entire datapath. The *bitop* multiplexer is then used to output the result of the appropriate operation. Furthermore, we reused the 16×16 multiplier for shifting operations as described in Section 6.4. Supporting SHA-1 operations on prime-field ECC processors is therefore not an expensive issue regarding the additional amount of gates in the datapath.

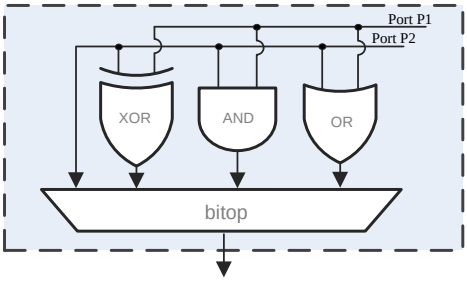


Figure 6.5: Logic gates of the datapath used for SHA-1.

6.2.5 8-bit Controller

A sophisticated two-layer approach has been necessary to efficiently implement the controlling of the cryptographic processor. The generation of control signals for various irregular algorithms and protocols, which require in total more than 800 000 clock cycles, is very complex. The highest layer of the controller comprises an 8-bit microcontroller with a highly optimized instruction set. It performs higher-level functions like protocol handling, point multiplication, and invocation of round functions for SHA-1 and AES due to its ability for looping and subroutine calls. The advantage of having a microcontroller is that it is very flexible because of extending the functionality by simple Assembler programming. Basically the microcontroller sets up and calls certain instructions which lie in a subsequent microcode ROM table (the second control layer). For the execution of such instructions, which can take up to 102 clock cycles, the start address in the microcode ROM has to be provided. While the microcode ROM provides instructions for the datapath and the memory via the instruction and the address decoder, the microcontroller can set up the next instruction. This avoids idle cycles of the datapath during execution of the algorithm.

Our proposed microcontroller is based on a Harvard architecture, i.e. program memory and data memory are separated. Such a design has the advantage that the program memory can have a different bit width than the data memory. The microcontroller is a Reduced Instruction Set Computer (RISC) supporting 32 instructions that have a width of 16 bits. The instructions are mainly divided into four groups: logical operations like XOR and OR, arithmetic operations like addition (ADD) and subtraction (SUB), control-flow operations like GOTO and CALL, and microcode instructions (MICRO). A list of the supported instructions is given in Table 6.1.

The main components of the microcontroller are the ROM, the register file, the program counter, the instruction-decode unit, and the Arithmetic Logic Unit (ALU). The ROM contains the program memory and has a size of 600×16 bits. The register file is the data memory of the microcontroller and consists of 11 registers with a width of 8 bits each. Instructions are executed within a two-stage pipeline that consists of a fetch and a decode/execute step. In the first stage, the instruction that is addressed by the 11-bit program counter is loaded from the ROM into the instruction-decode unit. In the second stage, the instruction is decoded by the instruction-decode unit and executed by the ALU, followed by updating the program counter. The program counter contains a call stack that allows up to three recursive subroutine calls. All instructions are executed within a single clock cycle, except the control-flow operations and the microcode operations. Control-flow operations require two clock cycles. The number of clock cycles required for a microcode operation is not fixed and depends on the instruction that is executed.

There are two types of registers in the register file of the microcontroller: special-purpose registers and general-purpose registers. The special-purpose registers involve an accumulator register for advanced data manipulation, a status register that gives information about the status of the ALU (e.g. carry bit after addition), and input/output (I/O) registers. The latter are used for accessing external devices like the memory unit or the ECDSA arithmetic unit via memory-mapped I/O. The I/O registers are also used for reacting on external events via busy waiting, since interrupts are not supported by the microcontroller. General-purpose registers are used for arbitrary data manipulations and temporarily storing data.

For implementing the microcontroller program, we developed a self-written instruction-set simulator and Assembler. Both programs are written in JAVA and allow a fast and easy way of program development. The simulator supports a single-step mode and gives access to the internal state of the microcontroller. This makes debugging and testing of the program very convenient. Moreover, optimizing the instruction set and adjusting parameters of the microcontroller like data bit width or call-stack size can be done much faster than in a direct hardware simulation. The output of the Assembler is a *VHDL* file that contains the implementation of the program ROM of the microcontroller. This file can then be used for hardware simulation and synthesis.

Table 6.1: Instruction set of the 8-bit microcontroller.

Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Flags[a]	Cycles	Description
AndLF	1	1	1	reg					constant								Z	1	AND constant to register
OrLF	1	1	0	reg					constant								Z	1	OR constant to register
MovLF	1	0	1	reg					constant								Z	1	move constant to register
XorLF	1	0	0	reg					constant								Z	1	XOR constant to register
GOTO	0	1	1	1	address													2	unconditional branch to relative address
CALL	0	1	1	0	address													2	call subroutine at relative address
BZ	0	1	0	1	address													1/2	jump to address if ZERO flag is set
BNZ	0	1	0	0	address													1/2	jump to address if ZERO flag is not set
MICRO	0	0	1	x	x	x	instr					addr						0..102	invokes a microcode instruction
MovFF	0	0	0	1	1	1	reg dst					reg src					Z	1	move register to register
AndFF	0	0	0	1	1	0	reg dst					reg src					Z	1	AND register to register
XorFF	0	0	0	1	0	1	reg dst					reg src					Z	1	XOR register to register
AddFF	0	0	0	1	0	0	reg dst					reg src					Z,C	1	ADD register to register
BTC	0	0	0	0	1	0	1	1	bit			reg						1/2	skip next instruction if bit is clear
BTS	0	0	0	0	1	0	1	0	bit			reg						1/2	skip next instruction if bit is set
BWC	0	0	0	0	1	0	0	1	bit			reg						1	stop program counter until bit is clear
BWS	0	0	0	0	1	0	0	0	bit			reg						1	stop program counter until bit is set
AddLW	0	0	0	0	0	1	1	1	constant								Z,C	1	ADD constant to ACC
SubLW	0	0	0	0	0	1	1	0	constant								Z,C	1	SUB constant from ACC
RetLW	0	0	0	0	0	1	0	1	constant									2	move const to ACC and return from subroutine
OrWF	0	0	0	0	0	1	0	0	x	x	T	reg					Z	1	OR register with ACC
RotlWF	0	0	0	0	0	0	1	1	1	1	T	reg					Z	1	rotate left through carry
RotrWF	0	0	0	0	0	0	1	1	1	0	T	reg					Z	1	rotate right through carry
ShlWF	0	0	0	0	0	0	1	1	0	1	T	reg					Z,C	1	shift left through carry
ShrWF	0	0	0	0	0	0	1	1	0	0	T	reg					Z,C	1	shift right through carry
DecWF	0	0	0	0	0	0	1	0	1	1	T	reg					Z,C,OV	1	decrement register
IncWF	0	0	0	0	0	0	1	0	1	0	T	reg					Z,C,OV	1	increment register
DecTWF	0	0	0	0	0	0	1	0	0	1	T	reg					Z,C,OV	1/2	decrement reg and skip instr if result is zero
IncTWF	0	0	0	0	0	0	1	0	0	0	T	reg					Z,C,OV	1/2	increment reg and skip instr if result is zero
RET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		2	return from subroutine
HALT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1		1	halt the machine in simulation
NOP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		1	no operation

[a] Flag Z indicates a zero value, C identifies a carry bit, and OV reports overflows.

6.2.6 Microcode Control

Next to the 8-bit microcontroller, which mainly handles the execution of different algorithms, we also implemented a tiny microcode controller. The microcode controller is basically used to execute different microcode instructions (MICRO) for fast ECC and AES computations. The controller is composed of a 7-bit pattern-operation counter, eight ROM tables, an instruction decoder, and an address decoder.

In order to reduce the footprint of our processor to a minimum, we stored all microcode instructions in dedicated ROM tables. In fact, we implemented eight ROM tables that provide different sizes. Thus, instructions which need less address bits can be stored in a smaller ROM table while others can be stored in larger tables. Table 6.2 lists the content of all implemented microcode ROM tables.

Table 6.2: The content of the implemented microcode ROM tables.

ROM Table	Patterns (Rows)	Bit Length (Columns)	Microcode Instructions
0	53	18 bits	Montgomery Multiplication, Montgomery Inversion, Loading of ROM Data
1	48	18 bits	Modular Addition, Montgomery Inversion
2	49	11 bits	SHA-1
3	99	14 bits	SHA-1
4	128	18 bits	Montgomery Multiplication, Modular Multiplication
5	101	18 bits	Modular Multiplication NIST Reduction
6	19	9 bits	AES
7	69	18 bits	Modular Subtraction, AES

In particular, each ROM table addresses up to 128 microcode patterns. One microcode pattern contains the opcode of the instruction, the memory read and write conditions, and the basis and offset address code of both memory ports, i.e. port P1 and port P2. The bit width of the patterns varies between 9 and 18 bits. AES, for example, needs only 9 bits for coding the instructions and addresses while modular operations for ECC need up to 18 bits.

The microcode controller works as follows. If a MICRO instruction is invoked by the microcontroller, several steps are executed. First, the microcode controller sets the pattern-operation counter according to the pattern size of the executed instruction. Furthermore, it sets the start address accordingly. In the second step, the proper ROM table is addressed and the first microcode pattern is executed. The instruction decoder decodes the instruction opcode and sets the

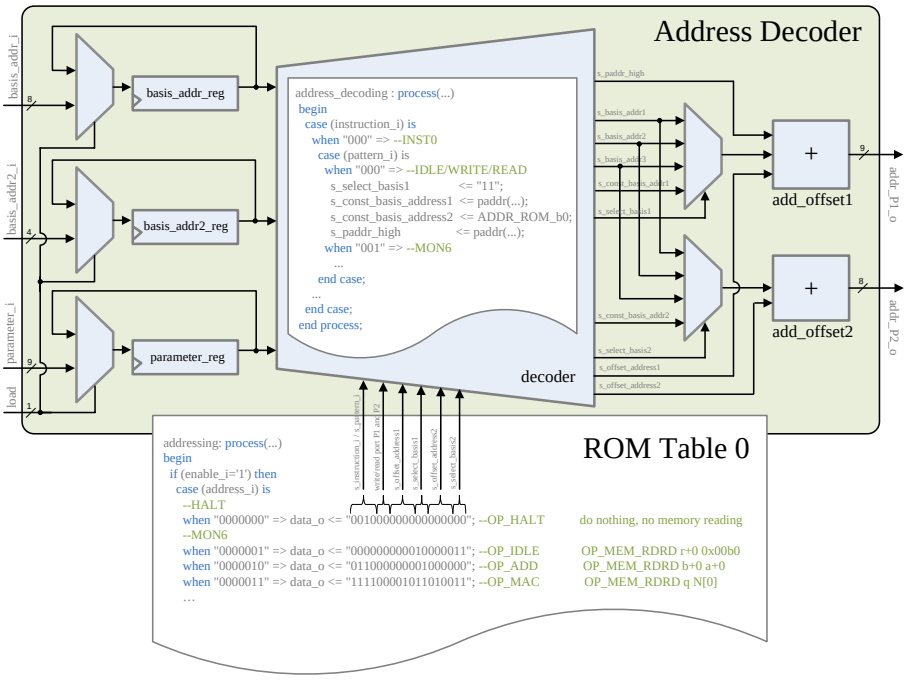


Figure 6.6: The microcode control reads an instruction from ROM table 0 and decodes the address patterns for port P1 and P2.

control signals of the datapath accordingly. The address decoder decodes the address code of the basis and offset addresses of both memory ports P1 and P2. After that, the next microcode pattern is executed in the next clock cycle. Figure 6.6 shows the address decoder that decodes the addresses of the patterns stored in ROM table 0. The decoded addresses for port P1 and P2 are then used by the memory unit to access the data used for that microcode instruction.

6.3 Arithmetic-Level Implementation

In the following, we focus on the arithmetic operations that are needed to perform ECC scalar multiplications. First, we will give an introduction into basic integer arithmetics such as addition, subtraction, and multiplication. Note that our processor supports only multiple-precision arithmetic instead of full-precision arithmetic operations. This reduces the area for the datapath (mainly the area of the ECDSA hardware multiplier and the needed registers to store the operands and results) because only 16-bit words are used instead of 192-bit words of the entire field size. Second, we will describe modular operations over prime fields used in ECC. We will describe modular addition, modular subtraction, and modular multiplication. For the modular reduction, we consider arbitrary primes as well

Algorithm 8 Multiple-precision addition.

Require: $a, b \in [0, 2^{Wt})$.
Ensure: $(\varepsilon, c) = a + b$.
1: (ε, C[0]) $\leftarrow$ A[0] + B[0].
2: **for** i **from** 1 **to** $t - 1$ **do**
3: (ε, C[i]) $\leftarrow$ A[i] + B[i] + ε.
4: **end for**
5: **Return** (ε, c).

Algorithm 9 Multiple-precision subtraction.

Require: $a, b \in [0, 2^{Wt})$.
Ensure: $(\varepsilon, c) = a - b$.
1: (ε, C[0]) $\leftarrow$ A[0] - B[0].
2: **for** i **from** 1 **to** $t - 1$ **do**
3: (ε, C[i]) $\leftarrow$ A[i] - B[i] - ε.
4: **end for**
5: **Return** (ε, c).

as Mersenne-like primes. Finally, we discuss Montgomery arithmetic operations that are used to provide efficient modular multiplication and inversion.

6.3.1 Integer Arithmetic

In the following, we will describe basic operations for multiple-word (multiple-precision) integers. Each multiple-precision integer is represented by a word-array structure, e.g. a = (A[t-1], ..., A[2], A[1], A[0]), where t represents the number of words.

Addition

Algorithm 8 shows the addition algorithm of two multiple-word (multi-precision) integers $a, b \in [0, 2^{Wt} - 1]$, where W denotes the number of bits of a word. The first step of the algorithm is to add each word A[i] and B[i] of the two operands a and b, where i represents the word index from 0 to t-1. The result of the word addition is stored in C[i] and the carry bit is stored in the variable $\varepsilon \in [0, 1]$. Note that the carry bit is added for each word and propagates through the entire integer addition. As a result of the addition, the integer $c \in [0, 2^{Wt})$ and the carry bit ε is returned [HMV04].

Subtraction

Algorithm 9 shows the subtraction algorithm of two integers $a, b \in [0, 2^{Wt})$. Instead of adding the words as in Algorithm 8, the words are subtracted. The borrow bit ε indicates negative results. Note that subtraction can be very efficiently implemented in hardware where simply the two's complement of one operand is added to the other operand. Therefore, hardware resources for addition can be efficiently reused.

It is also worth to note that addition and subtraction process each word of the operands only once which allows to overwrite one of the operands *in-place*. Thus, no additional variable (and memory) is needed to store the result c.

Multiplication

In the following, we will describe two integer multiplication methods that are often applied in practice: the *operand scanning* and the *product scanning* method. Both algorithms need t^2 single-precision multiplications and differ only in the way how to process the partial products and how the operands are loaded during the execution.

Algorithm 10 Schoolbook multiplication (operand scanning).

Require: Integers $a, b \in [0, 2^{Wt})$, $S \in [0, 2^{2W})$.
Ensure: $c = a * b$.
1: **for** i **from** 0 **to** $t - 1$ **do**
2: C[i] $\leftarrow$ 0.
3: **end for**
4: **for** i **from** 0 **to** $t - 1$ **do**
5: S $\leftarrow$ 0.
6: **for** j **from** 0 **to** $t - 1$ **do**
7: S $\leftarrow$ C[$i + j$] + A[i]*B[j]+(S $\gg W$).
8: C[$i + j$] $\leftarrow$ (S mod 2^W).
9: **end for**
10: C[$i + t$] $\leftarrow$ (S $\gg W$).
11: **end for**
12: **Return** (c).

Algorithm 10 shows the multiple-precision multiplication using the *operand scanning* method (often referred as the schoolbook method). It is mainly composed of two nested loops and a $2W$-bit accumulator variable S. In the inner loop, two load operations, two additions, and one store operation have to be performed. In particular, the variables B[j] and C[$i + j$] have to be loaded, added together with A[i], and stored in C[$i + j$]. The variable A[i] can be loaded outside the inner loop and kept in a register of the processor.

Algorithm 11 shows the multiple-precision integer multiplication using the *product scanning* method (often referred as the Comba method) [HMV04]. In contrast to the schoolbook method, it performs a store operation only in the outer loop structure. Thus, it needs fewer memory accesses than the schoolbook method. The size of the accumulator variable needs to be $3W$ bits.

6.3.2 Modular Arithmetic over Arbitrary Primes

Modular Addition

In general, modular addition and subtraction are the simplest arithmetic operations in ECC implementations. Due to the similar structure of the algorithms,

Algorithm 11 Comba multiplication (product scanning).

Require: Integers $a, b \in [0, 2^{Wt})$, $S \in [0, 2^{3W})$.
Ensure: $c = a * b$.

```
1:  S ← 0.
2:  for i from 0 to t − 1 do
3:     for j from 0 to i do
4:        S ← S + A[j]*B[i − j].
5:     end for
6:     C[i] ← (S mod 2^W).
7:     S ← (S ≫ W).
8:  end for
9:  for i from t to 2 ∗ t − 2 do
10:    for j from i − t + 1 to t − 1 do
11:       S ← S + A[j]*B[i − j].
12:    end for
13:    C[i] ← (S mod 2^W).
14:    S ← (S ≫ W).
15: end for
16: C[2t − 1] ← (S mod 2^W).
17: Return (c).
```

they need nearly the same amount of resources. Both addition and subtraction take between 5 and 15 % of the total execution time of one scalar multiplication.

Algorithm 12 shows the modular addition of the multiple-word integers $a, b \in [0, p-1]$, where p is the modulus with a size smaller than the word size of the operands, i.e. $p < 2^{Wt}$. After the integer addition, the intermediate result c is reduced. This can be typically done by subtracting the modulus p if the carry bit is set. An *if* condition, however, implies that the run time of the algorithm depends on the carry bit and the operands. In order to avoid side-channel leakages, it is recommended to provide a constant run-time either by an *if-then-else* condition [YJ00] or better by a regular structure [SS05].

Modular Subtraction

Algorithm 13 shows the modular subtraction of two multiple-precision integers a and b. The algorithm is similar to modular addition but performs subtractions instead of additions. In order to avoid negative results, the modulus p can be simply added to the negative intermediate result which guarantees that the final result c gets positive and smaller than p. The addition is performed only if the borrow bit $\varepsilon \in [0, 1]$ is set, which indicates a negative result. If the borrow bit is not set, a dummy operation is performed by addition of a zero value.

Algorithm 12 Modular addition.

Require: $a, b \in [0, p-1]$,
$\varepsilon, \varepsilon' \in [0, 1]$.
Ensure: $c = a + b \pmod{p}$.
1: Use Algorithm 8 to obtain (ε, c).
2: $(\varepsilon', \mathrm{C}[0]') \leftarrow \mathrm{C}[0]$ - P[0].
3: **for** i **from** 1 **to** $t-1$ **do**
4: $(\varepsilon', \mathrm{C}[i]') \leftarrow \mathrm{C}[i]$ - P[i] - ε'.
5: **end for**
6: **if** $(((\varepsilon \oplus 1) \ \wedge \ \varepsilon') = 1)$ **then**
7: **Return** (c).
8: **else**
9: **Return** (c').
10: **end if**

Algorithm 13 Modular subtraction.

Require: $a, b \in [0, p-1]$,
$\varepsilon, \varepsilon' \in [0, 1]$.
Ensure: $c = a - b \pmod{p}$.
1: Use Algorithm 9 to obtain (ε, c).
2: $(\varepsilon', \mathrm{C}[0]) \leftarrow \mathrm{C}[0] + (\mathrm{P}[0] * \varepsilon)$.
3: **for** i **from** 1 **to** $t-1$ **do**
4: $(\varepsilon', \mathrm{C}[i]) \leftarrow \mathrm{C}[i] + (\mathrm{P}[i] * \varepsilon) + \varepsilon'$.
5: **end for**
6: **Return** (c).

Modular Multiplication

Modular multiplication is one of the most expensive operations in ECC implementations. The number of invocations and the needed amount of clock cycles for one operation largely determines the performance of scalar multiplication. It is therefore advisable to spend high effort in the design of an efficient modular multiplication implementation.

Basically, there are many methods to perform modular multiplication. One of the most common ways is to perform an integer multiplication in the first step and a modular reduction in the second step. Common modular reduction methods for large integers are the Barrett reduction or the Montgomery reduction. By using special primes such as Mersenne-like primes, it is even possible to further increase the performance of modular reduction. Such special primes are considered in the next section.

Basically, we can distinguish between *separated* and *integrated* modular multiplication [KAJ96, Koç95]. Separated modular multiplications perform the multiplication first and apply the reduction afterwards. In this approach, the result of the multiplication is stored in a temporary variable r which is then reduced in a separated step, e.g.

$$\begin{aligned} r &\leftarrow a \times b, \\ c &\leftarrow r \pmod{p}. \end{aligned}$$

This approach needs additional memory to store the variable $r \in [0, 2^{2Wt})$.

The integrated (or interleaved) modular multiplication approach alternates between multiplication and reduction. There, partial products get reduced during the multiplication which avoids storing the double-sized result r and thus reduces the memory requirements significantly to the size of about the modulus $p \in [0, 2^{Wt})$ [Bla83, Slo85]. Many hardware implementations of ECC make use

of integrated multiplication to reduce the memory requirements, for example [Wol03, KAJ96, HWF08b, LSBV08].

In low-resource hardware designs, memory is one of the most expensive resources. In fact, more than 70 % of chip area for an ECC implementation is reserved for memory. Memory therefore largely determines the size of an ECC processor and the production costs of a chip. Hence, many implementations apply the reduction during the multiplication operation. This method has the advantage that no extra memory is needed to store the intermediate result. We therefore implemented a modular multiplication method with interleaved reduction for our processor. As a reduction method, we applied the fast NIST reduction algorithm described in the following.

6.3.3 Modular Arithmetic over the NIST P-192 Prime

In 1999, A. Solinas [Sol99] presented special primes that allow a very fast modular reduction of large integers. After that publication, the National Institute of Standards and Technology (NIST) recommended five different prime fields with such special moduli in 2000 [Nat00]. The proposed moduli have the advantage that they allow a very efficient modular reduction by simple additions. M. Brown et al. [BHHM01] performed several performance benchmarks in 2001. They used conventional workstations and reported a performance improvement of at least 2.5 compared to other existing reduction methods such as the Barrett reduction or the Montgomery reduction [HMV04].

The fast NIST reduction performs a modular reduction with additions only. For the smallest recommended prime $p = 2^{192} - 2^{64} - 1$, an integer $c = c_5 2^{320} + c_4 2^{256} + c_3 2^{192} + c_2 2^{128} + c_1 2^{64} + c_0$ can be reduced by simple additions

$$\begin{aligned} c \equiv\ & c_5 2^{128} + c_5 2^{64} + c_5 \\ & + c_4 2^{128} + c_4 2^{64} \\ & + c_3 2^{64} + c_3 \\ & + c_2 2^{128} + c_1 2^{64} + c_0 \pmod{p}, \end{aligned} \tag{6.1}$$

where c_i are represented as 64-bit integers. Note that the result is not fully reduced after the additions of the 64-bit words. An extra reduction step has to be performed in order to guarantee that the result c is indeed smaller than p. This can be done by repeatedly subtracting the modulus p from the almost reduced result c until it is smaller than p.

NIST P-192 Modular Addition and Subtraction

The NIST reduction can be used to reduce the carry bit of an integer addition. Instead of subtracting the modulus p from the intermediate result c of Algorithm 12, the carry bit $\varepsilon = 2^{192}$ can be simply added to the lower part of c using the congruence $2^{192} \equiv 2^{64} + 1 \pmod{p}$. Thus, c can be calculated by $c \leftarrow c + (2^{64} + 1) * \varepsilon$. However, after this addition, the result is smaller than

2^{Wt} but can be even larger than the modulus p so that an extra reduction has to be performed afterwards to guarantee $c \leq 2^{192} - 2^{64} - 1 < 2^{192}$. Note that this extra reduction is performed in extremely rare cases since the probability of occurrence is only

$$P(p \leq c) = \sum_{i=1}^{2^{64}} \frac{1}{2^{192}} = \frac{1}{2^{128}}. \tag{6.2}$$

For our processor, we decided to use the NIST reduction for modular addition and modular multiplication. Modular subtraction has been performed according to Algorithm 13. The implementation of the modular addition is as follows.

First, the microcontroller invokes the MICRO(INST_ADD) microcode instruction that performs an integer addition. The microcode for INST_ADD adds two 192-bit multiple-precision integers a and b within 19 clock cycles. For that, 19 microcode patterns have been implemented in ROM table 1. A possible carry bit is stored in the accumulator. After that, the fast NIST reduction is used to reduce the result. The NIST reduction has been implemented as a dedicated subroutine (NIST_RED). Within the subroutine, two microcode patterns are executed (i.e. INST_RED1 and INST_RED2) that have been implemented in ROM table 5. Both the INST_RED1 and INST_RED2 instruction take 12 clock cycles in total to reduce the result c to $c < 2^{192}$. The reason why we separated the reduction algorithm into two different microcode instructions lies in the fact that we would like to test if $c < p$ in a single clock cycle. This has been achieved as follows. The INST_RED1 instruction reduces the four least significant words of the result c by simply adding the carry bit ε. The remaining eight words are handled by the INST_RED2 instruction. During the INST_RED2 instruction, the datapath performs an AND operation on all bits of the remaining eight words (i.e. 128 bits). The resulting bit is stored in the Most Significant Bit (MSB) of the accumulator. If the MSB is one, the obtained result is larger than the modulus p and an extra reduction step has to be performed, otherwise it is zero. The microcontroller tests if the MSB of the accumulator is set by reading a dedicated memory mapped I/O register bit (CU_CARRY) which is directly connected to the ECDSA datapath.

The use of the NIST reduction for modular addition and subtraction has only an advantage when the reduction can be performed faster than a simple integer addition or subtraction. Otherwise, it is more advisable to apply Algorithm 12 and Algorithm 13. In our case, modular reduction can be performed faster than addition. It needs 12 clock cycles for the NIST reduction. Thus, a NIST P-192 modular addition needs 19+12+1=32 clock cycles. Note that 1 clock cycle is necessary to test the CU_CARRY bit signal of the datapath.

The integer subtraction has been implemented in ROM table 7 and needs 19 clock cycles. After the subtraction, the subroutine NIST_SUB_RED is called which adds the modulus p if the borrow bit ε is set. We also implemented a dedicated flag that enables a constant runtime for modular subtraction. If the *SPASaveReduction* flag is set, a dummy addition is performed if the borrow bit is zero.

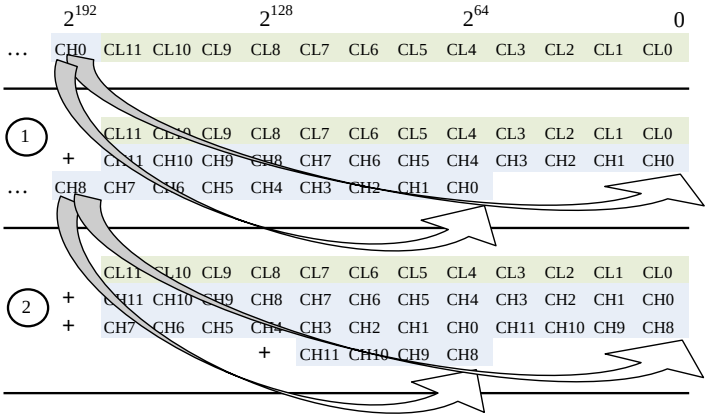


Figure 6.7: NIST P-192 reduction of the 384-bit result c.

NIST P-192 Modular Multiplication

As stated before, we implemented an integrated modular multiplication algorithm that reduces the result in an interleaved approach. The result is reduced by simple addition of individual partial products at different bit positions of the intermediate result. The algorithm is described in the following.

We decided to implement a multiplication algorithm in product-scanning form (Comba method, see Algorithm 11). Furthermore, we make use of the recommended NIST primes to allow fast reduction by simple additions. Thus, we make use of the congruency given in Equation (6.1). The first three lines reduce the higher part $c_H = c_5 2^{320} + c_4 2^{256} + c_3 2^{192}$. The result is then added to the lower part $c_L = c_2 2^{128} + c_1 2^{64} + c_0$.

Figure 6.7 illustrates the NIST reduction of the 384-bit result c in view of our 16-bit hardware architecture. There, the result c is represented in a multiple-word array structure which is composed of 24 elements:

$$c = (c_H | c_L) = (c_H[11], ..., c_H[0], c_L[11], ..., c_L[0]). \tag{6.3}$$

In order to reduce c, the higher part c_H can be simply added to the lower part c_L at the bit positions 0 and 63. This can be done because of the congruence $2^{192} \equiv 2^{64} + 1$. This results in a 256-bit integer which is still larger than 2^{192}. Thus, the remaining higher 64 bits can be also reduced by adding it at bit position 0 and 63. This results in an almost modulo-reduced 192-bit number.

Figure 6.8 shows the interleaved reduction of c_H. First, each partial product of c_H is calculated. Second, it gets reduced by the following partial-product additions. The lower four words of c_H are updated by the addition $(c_H[3]', ..., c_H[0']) = (c_H[3], ..., c_H[0]) + (c_H[11], ..., c_H[8])$. Next, the partial products $(c_H[7], ..., c_H[4])$ are added to the previous result $(c_H[3]', ..., c_H[0'])$. The result is stored in $(c_H[7]', ..., c_H[4]')$. Finally, the four most significant words are updated by subtracting $(c_H[3]', ..., c_H[0'])$ from $(c_H[7]', ..., c_H[4]')$ and by adding $(c_H[11], ..., c_H[8])$.

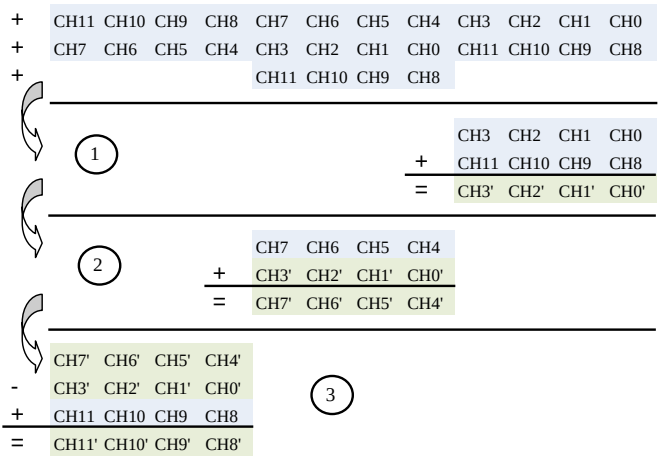


Figure 6.8: Reduction of c_H through simple additions of partial products.

The result is stored in $(c_H[11]', ..., c_H[8]')$. The subtraction is necessary because $(c_H[7], ..., c_H[4])$ has been already overwritten by $(c_H[7]', ..., c_H[4]')$ and has to be reconstructed.

The algorithm for the modular multiplication with interleaved NIST reduction is given in Algorithm 14. First, the higher part of the 384-bit result is calculated (line 1-8). Second, the higher part is reduced by subsequent additions to the lower part of c (line 9-20). Note that attention has to be paid for the carry propagation. After that, the lower part of the 384-bit result is calculated and added to the already reduced result (line 21-25). In line 26-30, the carry ε is reduced by adding ε to the accumulator variable S at word index 0 (line 21) and $t/3 = 4$ (line 26). Finally, a last reduction is performed in line 33-41 to reduce the final carry ε.

The modular multiplication has been implemented as a fully unrolled micro sequence. In particular, we separated the operation into two microcode instructions, i.e. INST_MUL1 (102 clock cycles) and INST_MUL2 (90 clock cycles). The entire modular multiplication including the final extra reduction needs 204 clock cycles. It needs t^2 single-precision multiplications to perform a modulo multiplication of two 192-bit numbers. No extra memory is needed to store the higher 192-bit part of the result. Instead, the result variable c is used to precompute the higher part and to reduce it before adding the lower 192-bit part. The algorithm is therefore highly suitable for resource-constrained devices.

6.3.4 Montgomery Arithmetic

Besides modular addition, modular subtraction, and modular multiplication (and squaring), ECC implementations also involve the operation of modular inversion. While elliptic-curve implementations using affine coordinates need an inversion for every group operation, projective coordinates have to compute the

Algorithm 14 Modular multiplication with interleaved NIST P-192 reduction.

Require: $a, b \in [0, p-1], S \in [0, 2^{3W}), \varepsilon \in [0, t + t/3 - 1]$.
Ensure: $c = a * b \pmod{2^{192} - 2^{64} - 1}$.

1: S $\leftarrow$ 0.
2: **for** i **from** 0 **to** $t - 1$ **do**
3: **for** j **from** $t - 1$ **to** i **do**
4: S $\leftarrow$ S + A[j]*B[$i + t - j$].
5: **end for**
6: C[i] $\leftarrow$ (S mod 2^W); S $\leftarrow$ (S $\gg W$).
7: **end for**
8: C[$t - 1$] $\leftarrow$ (S mod 2^W). S $\leftarrow$ 0.
9: **for** i **from** 0 **to** $t/3$ **do**
10: S $\leftarrow$ (S + C[i] + C[$i + 2t/3$]).
11: C[i] $\leftarrow$ (S mod 2^W); S $\leftarrow$ (S $\gg W$).
12: **end for**
13: **for** i **from** 0 **to** $t/3$ **do**
14: S $\leftarrow$ (S + C[i] + C[$i + t/3$]).
15: C[$i + t/3$] $\leftarrow$ (S mod 2^W); S $\leftarrow$ (S $\gg W$).
16: **end for**
17: **for** i **from** 0 **to** $t/3$ **do**
18: S $\leftarrow$ (C[$i + t/3$] - C[i] - S) + C[$i + 2t/3$].
19: C[$i + 2t/3$] $\leftarrow$ (S mod 2^W); S $\leftarrow$ (S $\gg W$) (mod 2).
20: **end for**
21: $\varepsilon \leftarrow$ S
22: **for** i **from** 0 **to** $t - 1$ **do**
23: **for** j **from** 0 **to** i **do**
24: S $\leftarrow$ S + A[$i - j$]*B[j].
25: **end for**
26: **if** ($\varepsilon = t/3$) **then**
27: S $\leftarrow$ S + C[i] + ε.
28: **else**
29: S $\leftarrow$ S + C[i].
30: **end if**
31: C[i] $\leftarrow$ (S mod 2^W); S $\leftarrow$ (S $\gg W$).
32: **end for**
33: $\varepsilon \leftarrow$ S
34: **for** i **from** 0 **to** $t - 1$ **do**
35: **if** ($\varepsilon = t/3$) **then**
36: S $\leftarrow$ S + C[i] + ε.
37: **else**
38: S $\leftarrow$ S + C[i].
39: **end if**
40: C[i] $\leftarrow$ (S mod 2^W); S $\leftarrow$ (S $\gg W$).
41: **end for**
42: **Return** (c).

modular inverse only once at the end of the scalar multiplication (to transform the projective coordinates back to affine coordinates). In fact, modular inversion is an expensive operation since it involves several trial divisions which are very expensive in both software and hardware implementations.

However, in 1985, P. Montgomery has shown that modular multiplication (and also inversion) can be performed more efficiently when trial division is replaced by divisions with two [Mon85]. In a binary-number representation, a division by two is a simple operation that can be performed by a simple right-shift operation. Nevertheless, in order to replace all operations by its Montgomery-arithmetic representatives, it is necessary to make some pre-computations and post-computations to transform the integers into the so-called Montgomery domain and vice versa. Let a be an integer in $[0, p-1]$, then a can be transformed into the Montgomery domain by

$$\tilde{a} = aR \pmod{p}, \tag{6.4}$$

where $\tilde{a}$ denotes the Montgomery-transformed integer a and R represents the Montgomery constant. R has to be chosen to be bigger and coprime to the modulus p, i.e. $R > p$ and $gcd(R, p) = 1$. The back transformation from the Montgomery domain into an integer is done by

$$a = \tilde{a}R^{-1} \pmod{p}. \tag{6.5}$$

In practice, often a Montgomery constant is chosen that has a power of two, e.g. $R = 2^{Wt}$. However, this number depends on the implementation and can be adjusted to obtain better performance for a specific architecture. But in general it is advisable to choose an R such that a division by R can be performed as efficient as possible to improve the performance of the modular operation. Moreover, both R and also R^{-1} are typically fixed and can therefore be stored in non-volatile memory or used as a hard-coded constant in ECC implementations.

Montgomery Multiplication

The Montgomery multiplication works as follows. Let $a, b \in [0, p-1]$ be two integers that are multiplied modulo a prime $p < 2^{Wt}$. We denote the multiplication function of Montgomery by $MontPro$ (Montgomery product) which is defined by

$$MontPro(\tilde{a}, \tilde{b}) = \tilde{a}\tilde{b}R^{-1} \pmod{p} = abR \pmod{p}. \tag{6.6}$$

Note that the algorithm requires as input the Montgomery-transformed integers $\tilde{a} = aR \pmod{p}$ and $\tilde{b} = bR \pmod{p}$ which can also be transformed using the $MontPro$ function, i.e.

$$\tilde{a} = MontPro(a, R^2) = aR^2R^{-1} \pmod{p} = aR \pmod{p}, \tag{6.7}$$

$$\tilde{b} = MontPro(b, R^2) = bR^2R^{-1} \pmod{p} = bR \pmod{p}. \tag{6.8}$$

The final result $\tilde{c} = MontPro(\tilde{a}, \tilde{b})$ can be transformed back into integer representation by

$$c = MontPro(\tilde{c}, 1) = (cR)R^{-1} \pmod{p} = c \pmod{p}. \tag{6.9}$$

Ç. Koç et al. [KAJ96] defined five different Montgomery-multiplication methods. Basically, they can be separated into *separated* and *integrated* methods. As it was described for modular multiplication, the separated approach performs a multiplication and reduces the result afterwards. The integrated approach interleaves these two operations. Furthermore, they can be separated into *coarse-grained* and *fine-grained* methods. *Coarse-grained* methods perform a reduction after the processing of a set of intermediate words. *Fine-grained* methods perform the reduction after each partial product. Finally, we can distinguish between operand, product, or hybrid scanning techniques as it is in the case of conventional modular multiplication. In the following, we will describe a Finely Integrated Operand Scanning (FIOS) method which has been used for our processor.

Algorithm 15 shows the implemented Montgomery multiplication [Mon85]. It takes two inputs $\tilde{a}, \tilde{b} \in [0, p-1]$ and calculates $\tilde{c} = MontPro(\tilde{a}, \tilde{b})$. As a Montgomery constant, we have chosen R to be $2^{W(t+1)} = 2^{16*13} = 2^{208}$. Thus, $t = 13$ loop iterations have to be performed (line 2). In the last round, all computations are performed with $\tilde{A}[t-1] = 0$ which avoids the need of additional memory for an extra 13th word. The upper bound of the loop-iteration result is $\tilde{c} < 2p$. A final subtraction reduces the result $\tilde{c}$ modulo p and a dummy subtraction is performed to provide a constant runtime (SPA resistance) [HQ00, Wal99, HMV04].

Table 6.3 lists all implemented microcode instructions for the Montgomery-multiplication operation. The first microcode instruction has been reused by other operations to clear 192-bit variables in RAM (set to zero). The MON6 operation has been implemented as a fully unrolled microcode sequence (48 lines of code). It also contains the pre-computation of S shown in line 4 in Algorithm 15. Shifts by W have been implicitly realized by an accumulator-shift operation to avoid additional clock cycles. The variable S has been hold in the accumulator while q has been stored in memory. The residue p'_0 has been pre-computed and

Table 6.3: Microcode instructions for Montgomery multiplication.

Instr. Name	Line Nr. in Alg. 15	Clock Cycles	Description
MON1/CLEAR	1	14	Initializes the variable $\tilde{c}$
MON2	3	5	Calculates the temporary variable q
MON3	10	3	Stores the final word in $\tilde{C}[t-1]$
MON4	2	2	Calculates q in the last round
MON5	12	1	Checks if the result $\tilde{c} \geq p$
MON6	4-9	48	Inner loop of the multiplication

Algorithm 15 Multiple-precision Montgomery multiplication (FIOS).

Require: $\tilde{a}, \tilde{b} \in [0, p-1], R = 2^{Wt}, p, p'_0 = -p_0^{-1} \pmod{2^W}$.
Ensure: $\tilde{c} = MontPro(\tilde{a}, \tilde{b}) = \tilde{a}\tilde{b}R^{-1} \pmod{p}$.
1: $\tilde{c} \leftarrow 0$.
2: **for** i **from** 0 **to** $t-1$ **do**
3: $q \leftarrow (\tilde{C}[0] + \tilde{A}[i]\tilde{B}[0])p'_0 \pmod{2^W}$.
4: $S \leftarrow (\tilde{C}[0] + \tilde{A}[i]\tilde{B}[0] + qP[0]) \gg W$.
5: **for** j **from** 1 **to** $t-1$ **do**
6: $S \leftarrow S + \tilde{A}[i]\tilde{B}[j] + qP[j]$.
7: $\tilde{C}[j-1] \leftarrow (S \bmod 2^W)$.
8: $S \leftarrow S \gg W$.
9: **end for**
10: $\tilde{C}[t-1] \leftarrow (S \bmod 2^W)$.
11: **end for**
12: **if** $(\tilde{c} \geq p)$ **then**
13: $\tilde{c} \leftarrow \tilde{c} - p$.
14: **else**
15: $\tilde{c} \leftarrow \tilde{c} - 0$.
16: **end if**
17: **Return** $(\tilde{c})$.

stored in ROM for the NIST prime p (i.e. $-p_0'^{-1} = -(0xFFFF)^{-1} = 0x0001$) and for the general prime n (i.e. $-n_0'^{-1} = -(0x2831)^{-1} = 0xCF2F$). The microcode instruction for the final subtraction operation used for the scalar multiplication has been reused by the microcontroller. In total, 785 clock cycles are needed to perform a Montgomery multiplication of two 192-bit numbers.

Montgomery Inversion

B. Kaliski [Kal95] proposed an algorithm to efficiently compute the Montgomery modular inverse in 1995. It takes an integer $a \in [0, p-1]$ and computes the Montgomery inverse, i.e.

$$MontInv(a) = \tilde{a}^{-1} = a^{-1}R \pmod{p}. \tag{6.10}$$

Note that the algorithm takes an integer a as an input and outputs the inverse of a in Montgomery-domain representation. However, if after a modular inversion a modular multiplication is followed, it is easy to implicitly transform the result back into an integer representation. The transformation from projective into affine coordinates in ECC implementations needs to compute $X_1 Z^{-1}$ (see Section 6.4 for more details) which can be done by evaluating

$$MontPro(X_1, MontInv(Z)) = X_1 Z^{-1} R R^{-1} = X_1 Z^{-1} \pmod{p}. \tag{6.11}$$

Algorithm 16 Montgomery inversion according to [Kal95].

Require: $a \in [0, p-1], R = 2^{Wt}$.
Ensure: $x_1 = a^{-1}R \pmod{p}$.
1: $u \leftarrow a$, $v \leftarrow p$, $x_1 \leftarrow 1$, $x_2 \leftarrow 0$.
2: **while** $v > 0$ **do**
3: **if** (v is even) **then** $v \leftarrow v/2, x_1 \leftarrow 2x_1$.
4: **else if** (u is even) **then** $u \leftarrow u/2, x_2 \leftarrow 2x_2$.
5: **else if** ($v \geq u$) **then** $v \leftarrow (v-u)/2, x_2 \leftarrow x_2 + x_1, x_1 \leftarrow 2x_1$.
6: **else** $u \leftarrow (u-v)/2, x_1 \leftarrow x_2 + x_1, x_2 \leftarrow 2x_2$.
7: **end if**
8: $k \leftarrow k + 1$.
9: **end while**
10: **if** ($u \neq 1$) **then**
11: **return** ("error. not invertible").
12: **end if**
13: **if** ($x_1 \geq p$) **then**
14: $x_1 \leftarrow x_1 - p$.
15: **end if**
16: **for** i **from** 0 **to** $k - Wt$ **do**
17: **if** (x_1 is even) **then**
18: $x_1 \leftarrow x_1/2$.
19: **else**
20: $x_1 \leftarrow (x_1 + p)/2$.
21: **end if**
22: **end for**
23: **Return** (x_1).

Note that one input operand of the Montgomery multiplication is in Montgomery domain and the other input operand is in integer representation. Thus, no back transformation has to be performed.

The implemented Montgomery-inversion algorithm is shown in Algorithm 16. First, all branches and loops have been implemented by the microcontroller. The loop index k in phase I (line 1-17) has been stored in an 8-bit general-purpose register of the microcontroller. Second, the right-shift operations (divisions by 2) have been realized using a dedicated microcode instruction (INV4 or INV10). INV11 handles the carry-bit propagation during shifting. The multiplications with 2 have been realized by simple additions. For that, we reused the already existing microcode instruction for addition as it has been also used during scalar multiplication. Third, we implemented a dedicated signal (CU_GET_LSB) from the datapath to the microcontroller to determine if a value is even or odd. Therefore, we implemented the INV6 microcode instruction, which loads either u or v from RAM and checks the LSB due to 0 (even) or 1 (odd). Furthermore, bigger or equal comparison has been done by using the already existing microcode instruction for subtraction. Table 6.4 lists all implemented instructions. Note

that INV1-3 and INV7-9 have been removed by optimizations and by replacement with already existing instructions. Our implementation needs 20 823 clock cycles for one modular inversion and requires mainly four working registers in RAM.

Table 6.4: Microcode instructions for Montgomery inversion.

Instr. Name	Line Nr. in Alg. 16	Clock Cycles	Description
INV4/INV10	1/2	13/12	Performs a right-shift operation
INV5	3	13	Zero check through logical OR operations
INV6	10	3	Load the variable u or v from RAM
INV11	2	2	Start right shift with carry

6.4 Algorithm-Level Implementations

In the following, we will describe the cryptographic algorithms used to perform ECDSA and AES. First, we will describe the implementation of SHA-1. After that, we will discuss the AES implementation in detail. Finally, we will focus on the ECDSA implementation.

6.4.1 The SHA-1 Algorithm

SHA-1 is an integral part of ECDSA and is used to hash digital messages. An input message m of any length $< 2^{64}$ bits is mapped to a 160-bit message-digest value. Basically, SHA-1 performs 32-bit operations on input blocks with a size of 512 bits. Thus, a message has to be subdivided into blocks of 512 bits. Shorter messages are padded accordingly. The main operations are bitwise logical AND, OR, XOR, NOT, left/right shift, and addition modulo 2^{32}. The input blocks are transformed within 80 rounds where five working variables (the *State*) are used, i.e. A,B,C,D, and E. Algorithm 17 shows the SHA-1 algorithm for one message block. For more information on the SHA-1 standard see FIPS-180-3 [Nat08].

For our ECDSA processor, we decided to sign messages with a fixed length of 16 bytes. This constraint allows us to reduce the SHA-1 implementation to only one 512-bit message block. In addition, the message padding can be implemented *a priori* by storing the length of the message in ROM. Thus, the 16 byte message can simply be copied into the RAM before signature generation. Message padding is done during the computation of ECDSA by copying the length of the message at the end of the input block W.

We implemented 13 different microcode instructions for SHA-1. Table 6.5 lists all implemented instructions with the corresponding line number of Algorithm 17. We integrated several implementation improvements. First, since line 11 and line 17 are the same, i.e. F $\leftarrow$ (B $\oplus$ C $\oplus$ D), we implemented only one microcode instruction (SHA7) that is invoked two times during the computation.

Algorithm 17 The Secure Hash Algorithm (SHA-1) [Nat02].

Require: 512-bit message block W; $H0, H1, H2, H3, H4, T, F, A, B, C, D, E \in [0, 2^{32} - 1]$.
Ensure: $h = \text{SHA-1}(W)$.
1: $A = H0; B = H1; C = H2; D = H3; E = H4$.
2: **for** i **from** 0 **to** 79 **do**
3: **if** $(i \geq 16)$ **then**
4: $\text{W}[i] \leftarrow \text{W}[i-3] \oplus \text{W}[i-8] \oplus \text{W}[i-14] \oplus \text{W}[i-16] \lll 1$.
5: **end if**
6: $\text{T} \leftarrow (\text{A} \lll 5) + \text{W}[i]$.
7: **if** $(i < 20)$ **then**
8: $\text{F} \leftarrow (\text{B} \wedge \text{C}) \vee (\overline{B} \wedge \text{D})$.
9: $\text{T} \leftarrow \text{T} + 0x5A827999$.
10: **else if** $(i < 40)$ **then**
11: $\text{F} \leftarrow (\text{B} \oplus \text{C} \oplus \text{D})$.
12: $\text{T} \leftarrow \text{T} + 0x6ED9EBA1$.
13: **else if** $(i < 60)$ **then**
14: $\text{F} \leftarrow (\text{B} \wedge \text{C}) \vee (\text{B} \wedge \text{D}) \vee (\text{C} \wedge \text{D})$.
15: $\text{T} \leftarrow \text{T} + 0x8F1BBCDC$.
16: **else**
17: $\text{F} \leftarrow (\text{B} \oplus \text{C} \oplus \text{D})$.
18: $\text{T} \leftarrow \text{T} + 0xCA62C1D6$.
19: **end if**
20: $\text{T} \leftarrow \text{E} + \text{F}$.
21: $\text{B} \leftarrow \text{B} \lll 30$.
22: $\text{E} \leftarrow \text{D}; \text{D} \leftarrow \text{C}; \text{C} \leftarrow \text{B}; \text{B} \leftarrow \text{A}; \text{A} \leftarrow \text{T}$.
23: **end for**
24: $H0 \leftarrow H0 + A$.
25: $H1 \leftarrow H1 + B$.
26: $H2 \leftarrow H2 + C$.
27: $H3 \leftarrow H3 + D$.
28: $H4 \leftarrow H4 + E$.
29: **Return** $(H0, H1, H2, H3, H4)$.

Second, instead of copying the values of the State variables (A,B,C,D,E) and a temporary variable T as shown in line 22, we simple rotated the addressing of the variables. Thus, no additional clock cycles are needed and the addresses get shifted by the microcontroller in every loop iteration. Third, all shift operations have been realized by multiplication. A left shift by one (line 4) is a simple multiplication with the constant 2 (stored in ROM at address ADDR_ROM_2 = 0xc4), a shift by five (line 6) is a multiplication by 32 (ADDR_ROM_32 = 0xc5), and a shift by 30 (line 21) is realized by a multiplication with 16 384 (ADDR_ROM_16384 = 0x0c6). Thus, no dedicated shift operation is necessary in the datapath of the processor. Fourth, the constants K0...K4 and the initial values for the chaining variables H0...H4 are stored in ROM and are loaded by

the microcode instruction SHA1. Fourth, the round loop is done by the microcontroller which also performs the branching at certain loop indices. Since the microcontroller can prepare the loop-index calculation during the execution of a microcode instruction, additional clock cycles are saved. In total, our processor needs 3 639 clock cycles to hash a 512-bit message.

Table 6.5: Microcode instructions for SHA-1.

Instr. Name	Line Nr. in Alg. 17	Clock Cycles	Description
SHA1	1	11	Loads the constants $H0...H4$ from ROM to RAM
SHA2	1	16	Initializes the variables (A...E)
SHA3	4	14	Instructions for loop index larger than 16
SHA4	6	8	Shifting A $\ll 5$ and adding W[i] to T
SHA5	8	9	Instructions for loop index smaller than 20
SHA6	9	3	Updating variable T by adding K0 (0x5A827999)
SHA7	11	7	Instructions for loop index smaller than 40
SHA8	12	3	Updating variable T by adding K1 (0x6ED9EBA1)
SHA9	14	13	Instructions for loop index smaller than 60
SHA10	15	3	Updating variable T by adding K2 (0x8F1BBCDC)
SHA11	18	3	Updating variable T by adding K3 (0xCA62C1D6)
SHA12	20-21	11	Updating variable T $\leftarrow$ E + F and B $\leftarrow$ B $\ll 30$
SHA13	24-26	9	Updating chaining variable H0, H1, and H2
SHA14	27-28	12	Updating chaining variable H3 and H4

6.4.2 The AES Algorithm

AES is a symmetric block cipher that was standardized by NIST [Nat01] in the year 2001. Basically, AES can be used with different key sizes, i.e. 128, 192, and 256 bits. The 128-bit version of AES (AES-128) takes blocks of input data and performs an encryption or decryption operation using 10 rounds. Within one round, several operations are performed that transform the input blocks inside the so-called *State*. The *State* is composed of 4×4 bytes, i.e. 128 bits. The performed operations are SubBytes, ShiftRows, MixColumns, and AddRoundKey. The SubBytes operation substitutes each byte of the *State*. This non-linear operation is typically realized as a look-up table (S-box) or calculated online during encryption or decryption. The ShiftRows operation rotates each row of the *State* while the MixColumns transforms each column of the *State* by a multiplication with a fixed polynomial. The AddRoundKey operation performs an XOR operation on every byte in the *State* and the corresponding key byte. A pseudo-code algorithm for AES-128 encryption is given in Algorithm 18. For a detailed description about AES see the FIPS-197 [Nat01] standard.

For AES, we implemented 11 microcode instructions. Table 6.6 lists all implemented instructions. The first two instructions AES1 and AES2 are used to initialize the AES *State* and to load the private key from the EEPROM.

Algorithm 18 Pseudo-code of AES-128 Encryption.

Require: message m, 4×4 bytes $State$, key w, $Rcon$ constant
Ensure: $c = enc(m)$
1: $State \leftarrow m$.
2: $w \leftarrow$ load_from_EEPROM(key).
3: AddRoundKey($State$, w).
4: **for** $r = 0$ to 9 **do**
5: $w \leftarrow$ SubBytes(Rot(w)) $\oplus$ $w \oplus Rcon$.
6: $w \leftarrow$ SubBytes(Rot(w)) $\oplus$ w.
7: $w \leftarrow w \oplus$ Rot(w).
8: $State \leftarrow$ SubBytes($State$).
9: $State \leftarrow$ ShiftRows($State$).
10: $State \leftarrow$ MixColumns($State$, r).
11: $State \leftarrow$ AddRoundKey($State$, w).
12: **end for**
13: **Return** $c = State$.

AES3-5 are responsible for the key expansion that is performed in each round. AES6 implements the SubBytes and ShiftRows operation. AES7-9 implements the MixColumns and AddRoundKey operation. AES10 makes a final conversion from 8-to-16-bit word size of the processor.

Due to the fact that our processor has an 16-bit architecture and that AES has been implemented with only 8 bits, we decided to use the remaining 8 bits to store dummy values. These values can be effectively used to perform dummy operations which increase the security of our processor in terms of side-channel attacks. Thus, all AES instructions can be performed on either the real or dummy values. Instead of a 16×8-bit AES *State*, we implemented a 16×16-bit *State* where the 8 bits are used to store the real *State* and the other 8 bits store a dummy *State* with random values ($r_0 ... r_{15}$).

Figure 6.9 shows the processing of the first AES *State*. The first operations are SubBytes and ShiftRows which transform the *State* column-wise, i.e. four byte blocks. First, each byte of the *State* is loaded and substituted by the S-box unit. In order to implicitly shift the bytes of the *State* for the ShiftRows operation, we simply addressed the appropriate bytes in the *State*. Thus, each byte of a column is addressed accordingly (address 0, 5, 10, and 15 in our example) and substituted afterwards. The result is stored in the accumulator of the datapath. After that, the bytes are loaded from the accumulator and processed by the MixColumns operation. The output is then XORed with the key within the AddRoundKey operation. Finally, the result is stored in the AES *State* in the RAM. After we processed the bytes of a column without the ShiftRows operation (in fact the bytes have not been shifted before MixColumns), we stored the resulting bytes in the correct position within the *State*. To avoid overwriting of data, we simply swap the values of the real and dummy *State*. Thus, after the

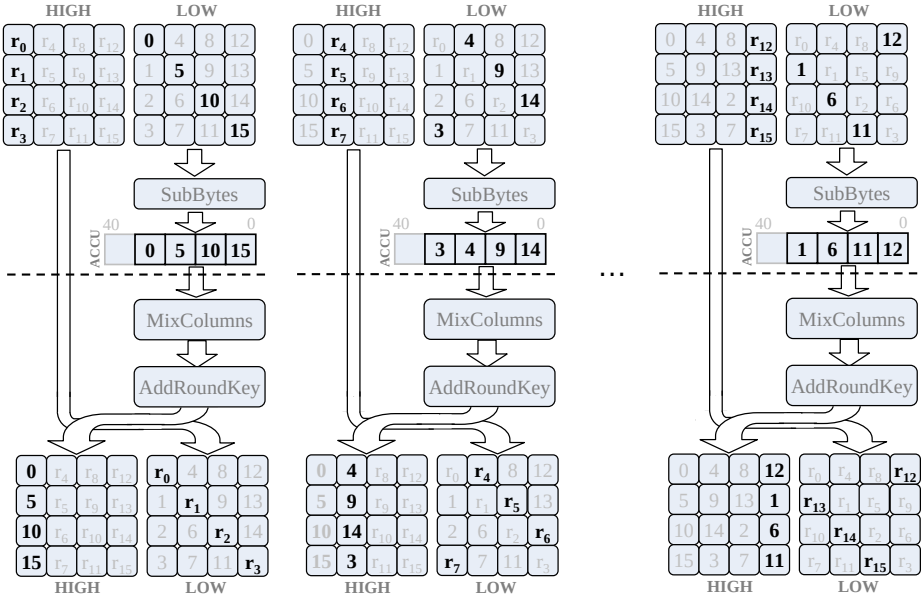


Figure 6.9: The processing of the *State* in the first AES round.

first round, the dummy values are stored in the lower 8 bits and the real values are stored in the higher 8 bits of the *State*.

In order to make the implementation less attractive to side-channel attacks, we integrated several countermeasures described in the following. As a first countermeasure, we integrated dummy operations before and/or after the actual AddRoundKey operation. 16 dummy operations are performed in total where the actual operation is widespread over 17 different locations in time.

As a second countermeasure, we randomized the processing of the bytes in the AES *State* which is often referred as *byte-shuffling* countermeasure. For this, we randomized the byte position after the SubBytes operation. The transformed bytes are stored in the accumulator with a random offset. After MixColumns and AddRoundKey transformation, the offset is incorporated through the right addressing of the *State*.

As a third countermeasure, we added 16 dummy rounds to the actual rounds. In fact, we performed dummy rounds only in the first and second round and the last two rounds of AES. This has its reason in the fact that side-channel attacks need to target intermediate values which can be generated by a model with less computational effort. Targeting intermediates of higher rounds would increase the computational effort significantly to generate values for each possible key. It is therefore sufficient to consider only the first and last two rounds of AES to obtain an appropriate protection.

All implemented countermeasures are commonly used in practice and provide a state-of-the-art protection for cryptographic devices. For a more detailed

description about dummy operations and shuffling (*hiding* techniques), see the work of S. Mangard et al. [MOP07].

Table 6.6: Microcode instructions for AES.

Instr. Name	Line Nr. in Alg. 18	Clock Cycles	Description
AES0	1	2	Copies the input data to the State
AES1	2	26	Loads the secret key from EEPROM to RAM
AES2	3	2	Performs the AddRoundKey operation
AES3	5	3	Key-expansion operation: SubBytes, XOR, Rcon
AES4	6	6	Key-expansion operation: SubBytes and XOR
AES5	7	2	Key-expansion operation: XOR
AES6	8-9	3	SubBytes and ShiftRows operation
AES7	10-11	4	MixColumns and AddRoundKey operation
AES8	10-11	4	Last round (AddRoundKey w/o MixColumns)
AES9	10-11	16	Last round operation (Shift accumulator)
AES10	13	3	Copy ciphertext to output (8-to-16 bit conversion)

6.4.3 Elliptic Curve Digital Signature Algorithm

ECDSA has been implemented in four phases. In the first phase, all variables for ECDSA are initialized. This phase includes the hashing of the message m, the generation of the ephemeral key k, and the randomization of the point coordinates. In the second phase, the scalar multiplication is performed which is in fact the most time-consuming operation. After that, the resulting point is checked if it is still a valid point on the elliptic curve. This operation increases the security level of our implementation since it evaluates the process due to errors and faults. In the last phase, the message is signed in a final signing-process step. All phases are now described in a more detail.

Initialization Phase

Table 6.7 shows the schedule of the RAM content during the initialization phase. The first two columns represent the RAM location in steps of 192 bits. Each column shows the RAM content after a specific operation. At address $0x00$, the RAM already holds a seed value that has been generated after power up of the processor, i.e. XREG (see Section 6.5.1 for a detailed description of the generated XREG value). The first operation for ECDSA is the hashing of the input message m. The 512-bit message is stored at address $0x48$, the 160-bit result is then stored at address $0x0C$. The next two operations generate two 192-bit random numbers. The first random number k is used as ephemeral key in ECDSA and is stored at address $0x18$. The second random number λ is used to randomize the projective coordinates of the initial base point P (see Chapter 4 or [Cor99] for a detailed description of the RPC countermeasure). Note

that we implemented a configuration flag in VHDL that indicates if the RPC countermeasure should be enabled or disabled. The fourth and fifth operations load the x-coordinate of the base point $x_P = (P, 1)$ and the doubled base point $x(2P) = (X_{2P}, Z_{2P})$ from ROM and store them at address $0x60$ (=x_P) and $0x48$ (=X_{2P}) and $0x54$ (=Z_{2P}). After that, the coordinates of P get randomized if the RPC countermeasure is enabled. This is done by a simple finite-field multiplication, i.e. λx_P (for this multiplication, the x-coordinate of P is loaded from ROM to allow an *in-place* operation). The result is stored at address $0x60$. If the countermeasure is disabled, the random value λ is overwritten by 1. The projective point $X_1 = (\lambda x_P, \lambda)$ is then stored at address 0x60 and 0x24 and the point $X_2 = (X_{2P}, Z_{2P})$ at address 0x48 and 0x54. Next, we applied the Co-Z technique described in Chapter 5 and stored the coordinates (X_1, X_2, Z) at address $(0x24, 0x30$, and $0x3C)$. Finally, we performed the scalar multiplication described in the following.

Table 6.7: Content of RAM before ECC scalar multiplication.

i	RAM addr.	hash msg	gen. k	gen. λ	load P	load $2P$	RPC	apply Co-Z	ECC kP
0	0x00	XREG	XREG	XREG	XREG	XREG	XREG	XREG	XREG
1	0x0C	**h(m)**	$h(m)$	$h(m)$	$h(m)$	$h(m)$	$h(m)$	$h(m)$	$h(m)$
2	0x18		**k**	k	k	k	k	k	k
3	0x24			λ	λ	λ	λ[a]	$\mathbf{X_1}$	$\mathbf{X_1}$
4	0x30							$\mathbf{X_2}$	$\mathbf{X_2}$
5	0x3C							$\mathbf{Z}$	$\mathbf{Z}$
6	0x48	m				$\mathbf{X_{2P}}$	X_{2P}	X_{2P}	*used*
7	0x54	m				$\mathbf{Z_{2P}}$	Z_{2P}	Z_{2P}	*used*
8	0x60	m			$\mathbf{x_P}$	x_P	$\lambda\mathbf{x_P}$[a]	λx_P	*used*
9	0x6C								*used*

[a] If the RPC countermeasure is disabled, the value λ is overwritten by 1 at address $0x24$ and x_P is not multiplied with λ at address $0x60$ to obtain $(x_P, 1)$ instead of $(\lambda x_P, \lambda)$.

ECC Scalar Multiplication

We applied the Montgomery ladder as scalar multiplication method. We further decided to apply Algorithm 4 described in Chapter 5. The formula provides out-of-place differential addition-and-doubling in projective co-Z coordinates. One point addition and doubling can be evaluated using $11\mathsf{M} + 4\mathsf{S} + 14\mathsf{add} + 1a + 1b4$ and 7×192-bit of memory in RAM. The constants $x_D, a, 4b$ have been stored in ROM and are directly loaded during the finite-field computations. Algorithm 19 shows the implemented scalar-multiplication algorithm.

After scalar multiplication, we perform a check if the point is still a valid point on the elliptic curve. For this, we need to recover the projective Y coordinate of the resulting point X_0. We applied Algorithm 5 given in Chapter 5 to recover the coordinates (X_0, Y_0, Z_0). The formula needs $8\mathsf{M} + 2\mathsf{S} + 8\mathsf{add} + 1a + 1b4$ and

Algorithm 19 ECC scalar multiplication using the Montgomery ladder in the Co-Z coordinate system.

Require: Base point $P = (x_P, y_P) \in E(\mathbb{F}_{p192}), k \in [1, n-1]$, random λ
Ensure: $Q = kP$, where $Q = (x_Q, y_Q) \in E(\mathbb{F}_{p192})$
1: $(X_0, Z_0) \leftarrow (\lambda x_P, \lambda)$.
2: $(X_1, Z_1) \leftarrow Dbl(P)$
3: $X_0 \leftarrow X_0 \cdot Z_1$, $X_1 \leftarrow X_1 \cdot Z_0$, $Z \leftarrow Z_0 \cdot Z_1$
4: **for** $i = 190$ **downto** 0 **do**
5: $\quad (X_{k_i \otimes 1}, X_{k_i}, Z) \leftarrow$ Algorithm 4 $(X_{k_i}, X_{k_i \otimes 1}, Z)$
6: **end for**
7: $(X_0, Y_0, Z_0) \leftarrow$ Algorithm 5 (X_0, X_1, Z, P)
8: **if** $Z'_0(Y'^2_0 - bZ'^2_0) \neq X'_0(X'^2_0 + aZ'^2_0)$ **abort.**
9: $x_Q \leftarrow X_0 \cdot Z_0^{-1}$
10: **Return** (x_Q).

can be evaluated with 7×192 bits of RAM. Note that we have to store also the y-coordinate of the base point in ROM, i.e. y_D.

After the curve equation check, the projective coordinates (X_0, Z_0) are transformed back into affine coordinates by applying a finite-field inversion and multiplication, i.e. $x_Q \leftarrow X_0 \cdot Z_0^{-1}$.

Point-Validity Check

After scalar multiplication, we validated if the resulting point (X_0, Y_0, Z_0) is still a point on the elliptic curve. This check (line 8 in Algorithm 19) is done by evaluating the elliptic-curve equation

$$Z'_0(Y'^2_0 - bZ'^2_0) = X'_0(X'^2_0 + aZ'^2_0). \tag{6.12}$$

The check has been proposed by N. Ebeid and R. Lambert [EL09] to prevent fault attacks on the x-coordinate only Montgomery ladder such as proposed by P. Fouque et al. [FLRV08] or M. Otto [Ott05]. Note that the check is done with projective coordinates to avoid a costly field inversion. The check is necessary to increase the security level of our processor. In fact, many fault attacks which target the processing of the scalar multiplication can be prevented [BMM00, CJ05, FGM+10].

The Signing Process

The final signing process of the ECDSA signature scheme has been realized in several steps. Note that all operations are now performed modulo the general prime n. First, the final value $\bar{x}$ of the scalar multiplication is tested to be zero or larger than the modulus n (the algorithm is aborted and an error message is returned otherwise). After that, the first part of the digital signature is stored

in RAM, i.e. r). Second, the following operation has to be performed to get the second part s of the signature:

$$s = k^{-1}(e + dr). \tag{6.13}$$

However, as shown in Chapter 4, this allows an attack to be performed on intermediate values during the private key multiplication dr. Since r is a known value (it is in fact a part of the digital signature) and d is a static value, an adversary is able to construct a model guessing intermediates of the multiplication. A DPA attack would therefore succeed and reveal the private key.

Thus, instead of evaluating Equation (6.13), we re-arranged the equation to prevent the attack described in Chapter 4. By evaluating

$$s = k^{-1}e + (k^{-1}r)d, \tag{6.14}$$

the fixed private key d gets multiplied by an unknown random value $k^{-1}r$. An adversary is therefore not able to perform a (first-order) DPA attack because $k^{-1}r$ is randomized and unknown in every signature generation. The additional cost for that countermeasure is one finite-field multiplication. For the evaluation of Equation (6.14), we applied one Montgomery inversion (see Algorithm 16 in Section 6.3) to invert k and three Montgomery multiplications: $k^{-1}e$, $k^{-1}r$, and $(k^{-1}e)d$. Note that the latter private-key multiplication requires the private-key operand to be in Montgomery-domain representation (see Algorithm 16 in Section 6.3 for more details).

Finally, the resulting digital signature is stored in RAM and consists of the 384-bit tuple (r, s).

6.5 System-Level Implementations

The following subsections present system and higher-level implementations. They include the generation of random numbers, protocol implementations such as entity authentication using AES or message authentication using ECDSA, as well as the certificate handling and the integration of our processor into existing public-key infrastructures.

6.5.1 Random-Number Generation

Random numbers are generated according to the FIPS 186-3 [Nat09] standard. The standard describes a hash-based pseudo-random number generator that can be realized with the SHA-1 algorithm. Our decision has mainly two reasons. First, the process of random-number generation is based on a standard specification and is considered to be cryptographically secure. Second, we already need a hash calculation for the message-digest calculation in ECDSA and we can simply reuse the implementation of the SHA-1 algorithm for that purpose. The prerequisite is to load a true-random seed from an external source into RAM at address 0x48. The random number is hashed and the message digest is stored

as a seed key (XKEY). This seed key is then used in any higher-level protocol to generate any random numbers needed to provide the cryptographic service.

6.5.2 Entity-Authentication Protocol using AES

AES has been integrated in a challenge-response protocol to provide entity authentication. We implemented the challenge-response identification mechanisms according to ISO/IEC 9798-2 [Int99]. Unilateral authentication can be performed by sending a random challenge to the prover. The prover encrypts the challenge and sends the result back to the verifier which can verify the authenticity of the prover. An RFID tag, for instance, can be authenticated to a reader by the following protocol:

Tag		**Reader**
$r_T \in_R \mathbb{Z}_n$		$c_R \in_R \mathbb{Z}_n$
	$\xleftarrow{\quad c_R \quad}$	
$c_T = E_K(c_R, r_T)$	$\xrightarrow{\quad c_T \quad}$	$(c'_R, r'_T) = D_K(c_T)$
		verify $c_R = c'_R$
		then accept, else reject

Figure 6.10: Tag to reader authentication according to ISO/IEC 9798-2 [Int99].

At first, the reader sends the challenge c_R to the tag. The tag generates a random number r_T and encrypts it together with the challenge of the reader. The ciphertext is transmitted back to the reader as a response. The reader accepts the tag if the decrypted data includes the generated challenge of the reader.

Tag		**Reader**
$r_T \in_R \mathbb{Z}_n$		$r_R \in_R \mathbb{Z}_n$
	$\xrightarrow{\quad r_T \quad}$	
	$\xleftarrow{\quad c_R \quad}$	$c_R = E_K(r_T, r_R)$
$(r'_T, r'_R) = D_K(c_R)$		
verify $r_T = r'_T$		
then accept, else reject		

Figure 6.11: Reader to tag authentication according to ISO/IEC 9798-2 [Int99].

Figure 6.11 shows the protocol for reader to tag authentication. First, the tag generates a random number r_T, which is retrieved by the reader. Second, the reader encrypts r_T together with an own generated random number r_R and sends it to the tag. The tag then decrypts the challenge and verifies the authenticity of the reader by comparing the decrypted data with the generated random number r_T.

The realization of the two described protocols given in Figure 6.10 and Figure 6.11 has been done by supporting three Application Protocol Data Unit

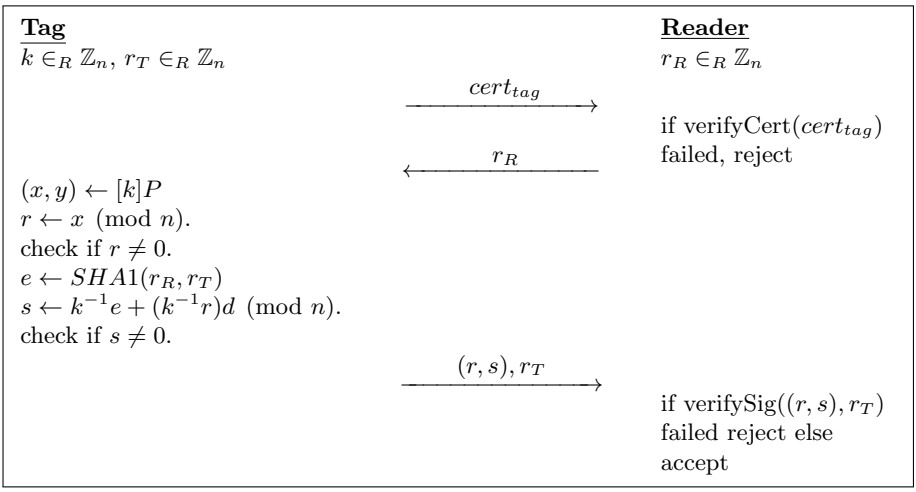


Figure 6.12: Message-authentication protocol using ECDSA.

(APDU) commands according to the ISO/IEC 7816-4 standard [Int95]. We implemented the commands *internal authenticate*, *get challenge*, and *external authenticate*. The *internal authenticate* command implements the protocol given in Figure 6.10. First, an 8-byte challenge has to be loaded into the RAM. Second, the processor generates a random number according to FIPS 186-2 [Nat00]. After that, it encrypts the 8-byte challenge c_R together with an 8-byte random number r_T using AES and stores the ciphertext in RAM to be transmitted back to the reader as a response.

Conversely, to authenticate an external device, we implemented the *get challenge* and *external authenticate* command (see [Int95] for more information). At first, the processor generates an 8-byte random number r_T which can be externally retrieved from RAM. The random number of the tag r_T and an 8-byte random number r_R of the reader are then encrypted using AES. Afterwards, the ciphertext c_R is sent back to the tag. Using the *external authenticate* command, the ciphertext is verified by decrypting the data in the memory. If the plaintext contains the same random number r'_T as previously generated r_T, an authentication flag is set in a status register of the microcontroller which can be queried by the external device.

6.5.3 Message-Authentication Protocol using ECDSA

ECDSA has been used in a challenge-response protocol to provide entity as well as message authentication according to the ISO/IEC 9798-3 [Int93] standard. Figure 6.12 shows the implemented protocol. Before starting the authentication process, the reader challenges the tag to get the public-key certificate $cert_{tag}$. After validation of the certificate, the reader chooses a random number r_R and sends it to the tag. After that, the tag digitally signs the challenge r_R of the

```
Certificate
  Version: 1
  Serial Number: 4660
  Signature Algorithm: ecdsaWithSHA1 (1.2.840.10045.4.1)
  Issuer: CN=TestCA
  Valid not before: Thu Feb 12 18:08:14 CET 2009
        not after: Tue Feb 12 18:08:14 CET 2019
  Subject: CN=14443A00,EMAIL=test@test.com
  SubjectPublicKeyInfo:
    Algorithm: ecPublicKey, NISTp192 (1.2.840.10045.2.1)
    SubjectPublicKey:
      03:32:00:04:62:B1:2D:60:69:0C:DC:F3:30:BA:BA:B6:
      E6:97:63:B4:71:F9:94:DD:70:2D:16:A5:63:BF:5E:C0:
      80:69:70:5F:FF:F6:5E:5C:A5:C0:D6:97:16:DF:CB:34:
      74:37:39:02
SignatureAlgorithm: ecdsa-with-SHA1 (1.2.840.10045.4.1)
Signature:
  30:35:02:18:1F:91:F5:89:8B:4F:C5:D3:47:D8:7C:F2:5D:8F:
  AE:53:6F:F7:39:3E:B2:D3:18:92:02:19:00:B4:F5:9A:F7:3B:
  13:80:48:B3:86:82:42:62:C8:23:57:7A:C5:A9:A6:B5:96:C2:
  D9
```

Figure 6.13: Structure of a standard X.509 certificate.

reader together with another random number r_T using ECDSA. First, it performs a scalar multiplication using the ephemeral key k and a fixed point P on the elliptic curve. The result r is then multiplied with the inverted ephemeral key k. Second, the private key d is multiplied to that (random) intermediate result. After that, the inverted ephemeral key k is multiplied to the hash e and added to the previous obtained result. Finally, the variables r and s represent the digital signature which is sent together with the random value r_T to the reader. The reader can then verify the signature of the message (r_R, r_T) and accept the tag if succeeded.

6.5.4 Public-Key Infrastructure and X.509 Certificates

In order to comply with existing Public-Key Infrastructures (PKI), we store a International Telecommunication Union (ITU-T) standard X.509 certificate of the used public key in the non-volatile memory of our low-resource processor. The certificate can be retrieved by any verifier by reading the data from the memory. Figure 6.13 shows the structure of a generated X.509 certificate. It contains attributes such as the version number, serial number of the certificate, issuer and subject identifier, validity, public-key data, and the signature of the certificate. The certificate is encoded using the Distinguished Encoding Rules (DER) which results in a binary representation of the given Abstract Syntax

Notation One (ASN.1).

In order to reduce the memory requirements for storing certificates on our processor, we applied two different techniques. First, we used *compressed* certificates which only store the x-coordinate of ECC points, e.g. the public key. The y-coordinate can be easily reconstructed by the verifier using one additional bit of information and by applying the curve equation. Note that this technique is known under *point compression* and is claimed by U.S. patent 6141420.

As a second memory-reduction technique, we stored only the variable part of the certificate in non-volatile memory of the prover. These are the public key, the serial number of the certificate, and the signature. The rest of the certificate can be reconstructed by the verifier which adds the remaining constant part of the certificate to obtain a valid X.509 certificate.

Both reduction techniques reduce the memory requirements for X.509 certificate by about 72 %. While a standard 192-bit certificate needs 268 bytes of memory, we only have to store 76 bytes for our memory-reduced certificate. A detailed evaluation of the implemented scenarios is given in [Hut09].

6.6 Synthesis Results

In the following, we give results of our implemented processor. We synthesized the processor for an Application Specific Integrated Circuit (ASIC) design and for an FPGA-based platform. First, we will present the results for the ASIC design. Second, we will focus on the FPGA implementation.

6.6.1 ASIC Synthesis Results

We implemented our processor in a 0.35 μm CMOS technology using a semi-custom design flow with Cadence RTL Compiler [Cad] as synthesis tool. The maximum clock frequency of the processor is 33 MHz. A photo of the chip die is shown in Figure 6.15.

The total chip area is 21 674 GEs and includes datapath, ROM, RAM macro, and controller for ECDSA, SHA-1, and AES (including microcontroller and microcode control). The area of the individual chip components are summarized in Table 6.8.

The datapath of our processor needs 3 604 GEs (2 205 GEs for ECDSA (where SHA-1 needs only 163 GEs) and 898 GEs for AES). In particular, the most resource-consuming component in the ECDSA datapath is the 16×16-bit multiplier which needs 1 619 GEs which is 73 % of the entire ECDSA logic or 45 % of the total datapath. Most area of AES is needed for the S-box and MixColumns operation, which need together 64 % of the AES datapath. The common 40-bit register needs 229 GEs.

The 8-bit microcontroller needs 1 703 GEs in total including register file (852 GEs), program counter (383 GEs) with 3-stage call-stack implementation, controller (252 GEs), and the ALU (200 GEs). The program ROM for ECDSA, SHA-1, and AES including the ISO/IEC 7816-4 [Int95] APDU handling and

Table 6.8: Area of chip components.

Component	**μm²**	**GE**
Datapath	198 271	3 604
ECDSA ALU	121 321	2 205
16 × 16 multiplier	89 053	1 619
AES ALU	49 413	898
S-box	20 293	368
MixColumns	11 484	208
Register	12 631	229
8-bit microcontroller	93 621	1 703
Registers	46 865	852
Program counter	21 112	383
Controller	13 905	252
ALU	11 011	200
Program ROM	173 082	3 146
Microcode controller	214 178	3 894
Microcode tables	120 702	2 194
ROM table 0	6 133	111
ROM table 1	8 954	162
ROM table 2	7 225	131
ROM table 3	15 907	289
ROM table 4	24 588	447
ROM table 5	16 125	293
ROM table 6	2 876	52
ROM table 7	13 723	249
Address decoder	78 897	1 434
Instruction decoder	14 323	260
ROM (ECC and AES constants, ...)	33 015	600
RAM macro (128x16-bit)	479 985	8 727
ECDSA+SHA1+AES Total	**1 192 070**	**21 674**

random-number generation takes 3 146 GEs in total. The microcode controller, in contrast, needs 3 894 GEs including the eight ROM tables, address and instruction decoder. The implemented ROM needs 600 GEs and stores constants for ECC and AES. Finally, we integrated a RAM macro which needs 8 727 GE. Note that the used RAM macro has not been optimized for low area and low power and leaves room for further improvements.

We also synthesized our processor without AES capabilities (i.e. datapath, program ROM, and microcode entries) resulting in 19 287 GEs. It shows that the integration of AES needs 2 387 GEs (for encryption and decryption) which is quite low compared to a stand-alone module integration. The same has been

Table 6.9: Cycle count of operations.

Component	Cycles
PRNG generation (4× SHA-1)	14 947
Point multiplication	753 393
Fault countermeasure	29 672
Final signing process	65 097
ECDSA Total	**863 109**
SHA-1	**3 639**
AES with shuffling	**4 529**

done for SHA-1. We removed any SHA-1 related material from the program ROM, microcode ROM tables, and datapath and synthesized the processor to evaluate the overhead of SHA-1. It shows that SHA-1 needs 889 GEs. The overhead of both algorithms is very low because most of the resources have been reused by our design such as the microcontroller, the RAM memory, and the datapath register.

Table 6.9 shows the clock-cycle distribution of the implemented components. In view of ECDSA, the point multiplication can be performed in 753 393 clock cycles. The entire ECDSA signing process takes 863 109 clock cycles including side channel and fault-attack countermeasures (RPC and point-validity check). The point-validity check needs 29 672 clock cycles which is 3.4 % of the total runtime. All numbers already include the generation of random numbers using four SHA-1 invocations that are 14 947 clock cycles (1.7 % of the total run-time). In addition, SHA-1 needs 3 639 clock cycles to hash a single message. AES with byte shuffling takes 4 529 clock cycles and 15 577 cycles with additional 10 dummy rounds.

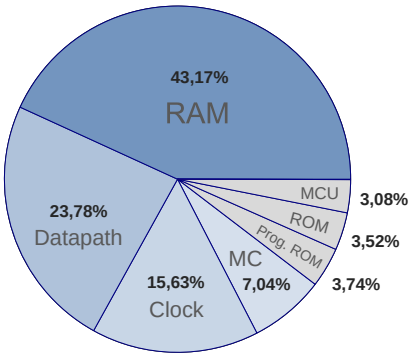


Figure 6.14: Power-consumption chart.

6.6.2 Power Simulation

We also simulated the total power consumption of our processor. The mean current of the circuit was simulated using Synopsis NanoSim which is 485 μA at 847 kHz and 3.3 V. This value includes the power consumption of the entire processor including microcontroller, ECDSA, SHA-1, and AES datapath, and memory. Note that this value can be decreased by using a lower supply voltage or a smaller process technology (for example CMOS 0.18 μm). In addition, about 43 % of the total power is consumed by the memory unit. By applying a low-power module, the power consumption of our processor can be further reduced. In fact, the highest power consumption is due to the used RAM macro followed by the datapath, clock tree, and microcode control circuit. A power-consumption chart is shown in Figure 6.14. It shows that the program ROM, the ROM for ECC constants, and the microcontroller circuit need almost no power

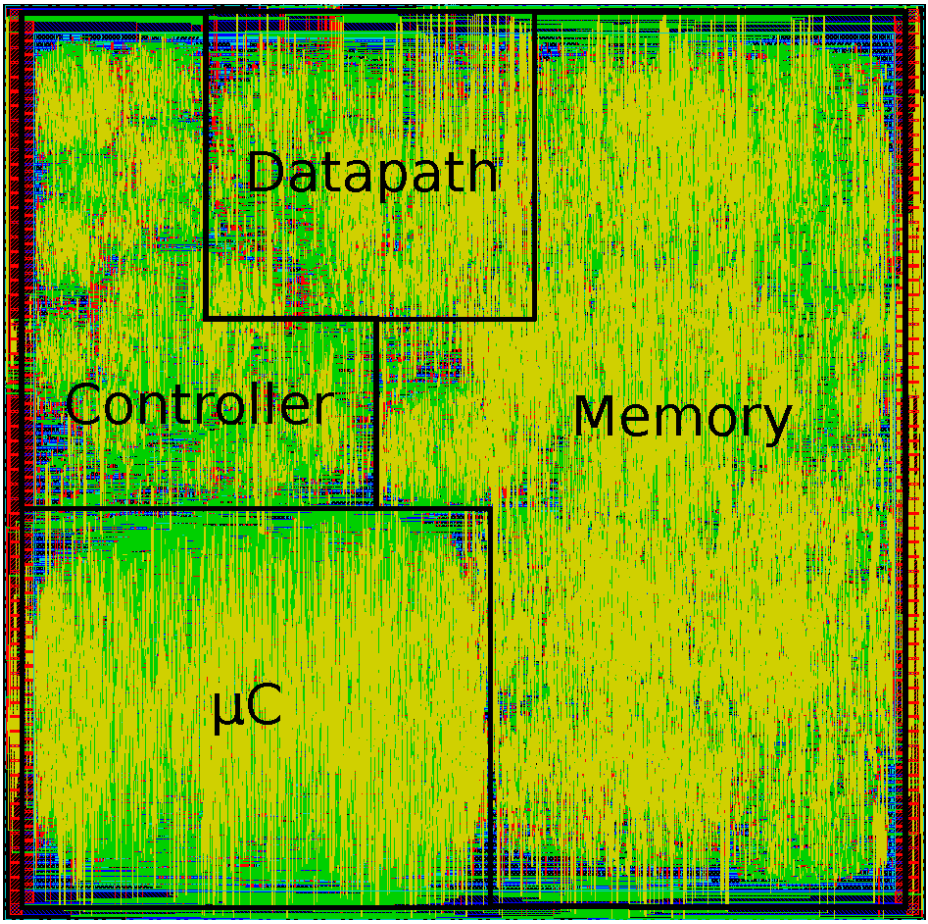


Figure 6.15: Chip layout of the processor.

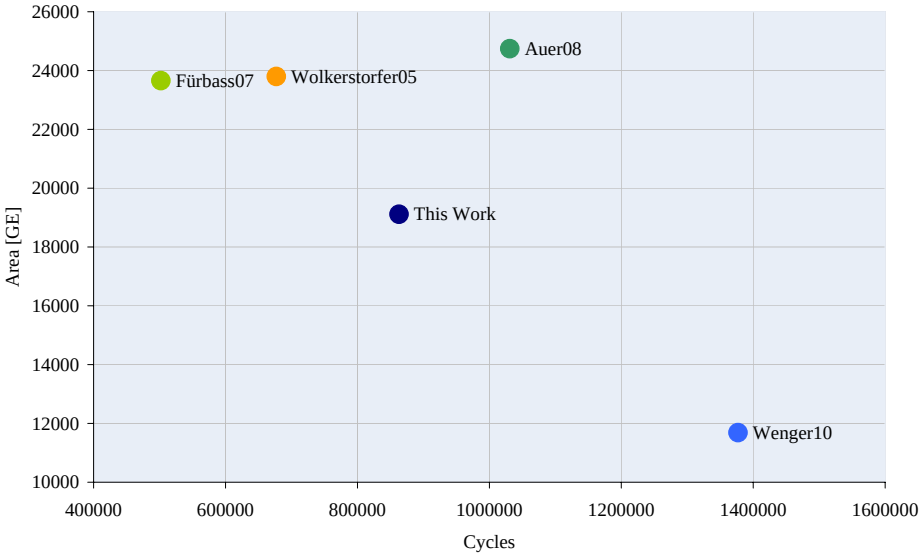


Figure 6.16: Comparison of different ECDSA implementations over $\mathbb{F}_{p192}$.

(around 3-4 %) of the total power consumption) while clock tree, datapath, and microcode control need almost 50 % of the total power consumption.

6.6.3 Comparison with Related Work

Table 6.11 gives a comparison of different ASIC implementations. We split the table into three main groups: ECC implementations over $\mathbb{F}_{p192}$, implementations over $\mathbb{F}_{2^m}$, and hardware/software co-design approaches. Note that a fair comparison is largely infeasible since the implementations differ in several ways. The given area sizes of the listed solutions differ due to the use of different elliptic curves, inclusion of RAM memory, implementation of countermeasures, protocol implementation, message digest generation (hashing), and the integration of a PRNG module.

In view of $\mathbb{F}_{p192}$, F. Fürbass et al. [FW07] and also J. Wolkerstorfer [Wol05] require around 23 700 GEs for the point-multiplication unit and A. Satoh et al. [ST03] need 29 655 GEs of chip area for the same size of prime-field arithmetic. E. Öztürk et al. [ÖSS04] need around 30 000 GEs for a co-processor over the prime field $\mathbb{F}_{(2^{167}+1)/3}$. E. Wenger [WFF11] needs only 11 686 GEs because of a single-port instead of a dual-port memory. Thus, their implementation is much slower which is 1 377 000 clock cycles. The implementation of A. Auer [Aue08] uses a dual-port memory and needs about 25 000 GEs and 1 031 000 clock cycles. T. Kern et al. [KF10] implemented a processor over $\mathbb{F}_{p160}$ using the SECG elliptic curve [Cer00b]. They need 18 247 GEs and 511 864 cycles to sign a message.

In Figure 6.16, all ECDSA implementations over the recommended NIST field

$\mathbb{F}_{p192}$ are shown. The x-axis represents the cycles and the y-axis represents the area in gate equivalents. It shows that our processor provides a good trade-off between area and speed.

In comparison with ECC implementations over binary fields, our implementation can compete with the work of S. Kumar and C. Paar [KP06]. Their solution is based on a similar field size of $\mathbb{F}_{2^{193}}$ which requires 19 048 GEs for point multiplication only. We also compare our solution with hardware/software co-design implementations of elliptic curves over $\mathbb{F}_{2^{191}}$. M. Koschuch et al. [KLW+06] implemented instruction set extensions on a synthesized 8051 Dalton microcontroller. Their architecture needs 29 491 GEs of total chip area. The work of H. Aigner et al. [ABHW04] provides only the results for the point-multiplication unit which takes around 25 000 GEs. Their design needs an external controller to perform ECDSA signature generation.

6.6.4 FPGA Synthesis Results

We also synthesized our processor on a Spartan-3 XC3S1000 FPGA from Xilinx. The implementation needs 7 RAM blocks to store memory related data of the EEPROM, RAM, ROM, and program ROM of the microcontroller. The processor further needs 2 806 slices including a digital front-end for an RFID-tag implementation and occupies one hardware multiplier for the multiply-accumulate unit in the crypto part. In particular, the digital front-end implements the standard ISO/IEC-14443 A (layer 1-4) [Int08a]. In our scenario, the processor is integrated in an RFID tag that communicates with a reader over a 13.56 MHz HF field. A detailed description of the overall RFID system can be found in [FAH+10].

Table 6.11 shows a comparison of our work to other ECC implementations over prime fields on FPGAs. C. McIvor et al. [MMM04] implemented a co-processor to perform a 256-bit scalar multiplication. Their implementation uses a 18×18-bit multiplier and includes a finite-field inversion algorithm. It needs 15 755 slices and 256 multipliers to perform a multiplication. G. Orlando and C. Paar [OP01] implemented a co-processor over $\mathbb{F}_{p192}$ using a high-radix Montgomery multiplier. They need 5 708 slices and 35 BRams to perform a scalar

Table 6.10: Comparison with other implementations of ECC.

Scheme	Device	Field	Area
Our work	**XC3S1000**	$\mathbb{F}_{p192}$[a]	**2 806 Slices / 7 BRams / 1 Mult**
ECC [MMM04]	XC2VP125	$\mathbb{F}_{p256}$	15 755 Slices / 256 Mults
ECC [OP01]	XCV1000E	$\mathbb{F}_{p192}$	5 708 Slices / 35 BRams
ECC [OBP02]	XCV1000E	$\mathbb{F}_{p160}$	6 055 Slices
ECC [VMG+10]	XC2VP30	$\mathbb{F}_{p160}$	1 832 Slices / 9 BRams / 2 Mults

[a] The numbers are for the entire processor including ECDSA, AES, and SHA-1.

multiplication. B. Ors et al. [OBP02] reported a co-processor over $\mathbb{F}_{p160}$ which needs 6 055 slices. The result of J. Vliegen et al. [VMG+10] needs only 1 832 slices and 9 BRams.

6.7 Summary and Conclusions

In this chapter, we presented results of a low-resource processor that leverages symmetric as well as asymmetric cryptography in one piece of silicon. We implemented the standardized algorithms AES and ECDSA and make several optimizations on the arithmetic, algorithmic, and implementation level. The outcome is an application-near solution that offers a large scale of cryptographic services such as entity and message authentication, non-repudiation, data integrity, and confidentiality. The implemented primitives allow the realization of different applications like proof of origin authentication of RFID-labeled goods, for instance. We based our design on a tiny microcontroller that uses a microcode-control approach to allow fast computations for ECC. The processor needs 19 287 GEs of chip area and signs a message within 859 188 clock cycles (a single point multiplication can be performed with 753 393 clock cycles).

As an outcome of the work, it has shown that a microcontroller facilitates the implementation of complex systems especially when several algorithms are implemented. First, microcontrollers provide the possibility to execute programs that have been written in Assembly-like languages. This reduces the complexity of the controlling and offers a familiar environment for developers. Second, they allow the simulation of programs without hardware synthesis which reduces the development time significantly. A microcode-control approach additionally provides fast computations that are needed for ECC implementations. Microcode instructions can be simply added or removed without modifications in the controlling such as it is in the case of dedicated finite-state machines. They are therefore more flexible and scalable. Third, microcontrollers reduce the area requirements since many resources of individual chip components can be shared. These resources are mainly for controlling and memory which constitute a large part in the total chip area.

We first combine AES and ECDSA in one piece of silicon. First, we implemented a stand-alone ECDSA implementation which needs 19 287 GEs. After that, we integrated AES-128 (encryption and decryption) additionally. It has shown that the integration of AES needs less than 2 400 GEs which is lower than existing stand-alone solutions based on finite-state machines (see Section 6.1 for related work). SHA-1 needs an overhead of about 900 GEs.

It has further shown that AES as well as ECDSA can be effectively protected against most attacks described in Chapter 2, Chapter 3, and Chapter 4. In view of RFID implementations, shuffling of the AES *State* as well as randomizing projective coordinates in ECC are very efficient and cheap in terms of chip area and execution time. Additional randomization of AES rounds needs an increased execution time and increases the security level in applications where time is a minor requirement. Elliptic-curve point-validity checks need the recovery of the

y-coordinate and require less than 5 % of additional area and execution time. Thus, they are especially cheap for processors that also verify ECDSA signatures in which case the y-coordinate is required for the verification.

Table 6.11: Comparison of different ECC-hardware implementations.

	Area [GE]	Cycles	Field	Features
This Work	**21 674**	**863 109**	$\mathbb{F}_{p192}$	**ECDSA, SHA-1, AES**
Wenger11 [WFF11]	11 686	1 377 000	$\mathbb{F}_{p192}$	ECDSA, SHA-1
Kern10 [KF10]	18 247	511 864	$\mathbb{F}_{p160}$	SECG curve, ECDSA, SHA-1
Auer08 [Aue08]	24 745	1 031 000	$\mathbb{F}_{p192}$	ECDSA, SHA-1
Fürbass07 [FW07]	23 656	502 000	$\mathbb{F}_{p192}$	ECDSA (no SHA-1, no RNG)
Wolkerstorfer05 [Wol05]	23 800	677 000	$\mathbb{F}_{p192}$	ECC
Öztürk04 [ÖSS04]	30 333	545 440	$\mathbb{F}_{(2^{167}+1)/3}$	ECC
Satoh03 [ST03]	29 655	4 165 000	$\mathbb{F}_{p192}$	ECC
Hein08 [HWF08b]	11 904	296 000	$\mathbb{F}_{2^{163}}$	ECC
Bock08 [BBD+08]	12 876	80 000	$\mathbb{F}_{2^{163}}$	ECC, DH, RNG
Lee08 [LSBV08]	12 506	302 457	$\mathbb{F}_{2^{163}}$	ECC, Schnorr
Kumar06 [KP06]	19 048	527 284	$\mathbb{F}_{2^{193}}$	ECC
Batina06 [BMS+06]	8 104	353 000	$\mathbb{F}_{2^{131}}$	ECC, without memory
Schroeppel02 [SBG+03]	191 000	93 000	$\mathbb{F}_{2^{178}}$	ECC, ElGamal, PRNG
Koschuch06 [KLW+06]	29 491	1 416 000	$\mathbb{F}_{2^{191}}$	ECC, 8051 μC
Aigner04 [ABHW04]	25 000	469 385	$\mathbb{F}_{2^{191}}$	ECDSA (no SHA-1, no RNG)

7 Conclusions

In this thesis, we examined the security of RFID devices in terms of practical implementation attacks and by designing an own cryptographic RFID processor. We structured the work into two main parts. The first objectives have been to evaluate the suitability of implementation attacks on passive RFID devices. During that part, we answered the general question if such low-resource devices are vulnerable to these attacks and if the threat even exists for such platforms. Next to these findings, we targeted a low-resource processor for RFID devices in the second part of the thesis. The processor has to leverage AES as well as ECDSA to provide most of the cryptographic services nowadays. The challenges in that hardware design have been to meet the fierce requirements of low area and low power while providing an appropriate level of security. In contrast to existing (finite-state based) solutions, we targeted a new approach by implementing a microcode-control together with a tiny microcontroller to hold the area requirements within limitations while providing a maximum of flexibility and scalability. In this context, we also decided to implement the ECC part of the processor over prime fields in order to evaluate the realizability of reducing the resource requirements of prime-field processors to a minimum.

The first part of the thesis focuses on implementation attacks. In Chapter 2, we presented practical side-channel attacks on passive RFID devices. Several setups have been proposed that allow the measurement of side-channel information. It turned out that passive tags leak information in several side channels which can be exploited. Best results have been obtained by measuring the electromagnetic emanation where the reader field was canceled out by appropriate filtering techniques.

Chapter 3 presented fault attacks on passive RFID tags. We presented several fault-injection setups and performed different attacks on commercially-available

devices. It showed that they are susceptible against these kind of attacks and allowed the writing of faulty values into the memory while confirming the operation to be successful. Such implementation weaknesses can be explored by more sophisticated attacks to misuse an existing RFID system.

After that, we targeted public-key enabled RFID devices in Chapter 4. We first presented common public-key protocols for tag authentication and demonstrated an attack on the ISO/IEC 9798-3 entity-authentication protocol that uses ECDSA as a signature scheme. In particular, we performed power and electromagnetic side-channel attacks on a low-resource ECDSA hardware implementation. It showed that public-key enabled tags are vulnerable to these attacks by extracting the private key used during the signature generation. In addition, we show that any public-key implementation is vulnerable to this attack that implements a finite-field multiplication in multiple-precision arithmetic.

These findings led us to the second part of the thesis. This part focuses on the implementation of a low-resource hardware processor for passive RFID devices. We decided to implement both AES as well as ECDSA in one piece of silicon. First, we focus on the ECDSA implementation and presented new formulæ for ECC in Chapter 5. The formulæ provide differential addition and doubling using the Montgomery ladder and apply a shared common Z-coordinate technique to minimize the memory requirements. Furthermore, we are the first who present out-of-place formulæ for prime field arithmetics that outperform existing formulæ in literature.

Finally, we applied these formulæ in an ECDSA hardware implementation in Chapter 6. As already stated before, we applied a new concept where a microcode-control architecture and a tiny 8-bit microcontroller is used to provide high flexibility on the one side and fast execution of ECC operations on the other side. Furthermore, we are the first who combined AES together with ECDSA in one hardware implementation. Both algorithms share common resources such as memory and controlling. Moreover, we integrated several countermeasures against attacks presented in the first part of this thesis. It showed that randomization countermeasures need almost no resources which encourages the use in RFID implementations. Different checks of intermediate values and results prevent most fault attacks by needing less than 5 % of area and speed. As an outcome, we presented a cryptographic processor that provides an optimal trade-off between area, speed, and security compared to existing solutions in literature.

One conclusion of this thesis is that passive RFID devices are vulnerable to implementation attacks and need appropriate countermeasures. Even public-key cryptography enabled RFID tags have to be protected to prevent cloning and thus impersonation of tags. This is especially important in the field of anti-counterfeiting where products get labeled with RFID tags which provide cryptographic services to allow a verifiable (and maybe transferable) proof-of-origin. In such a scenario, adversaries might be able to extract the private key of such tags to forge the product. Side-channel attacks as well as fault attacks (and even combined attacks) pose a serious threat in that context that should be considered in future implementations.

Another outcome of this work is that a microcontroller architecture reduces the area requirements especially when several hardware components can share common resources. In our design, the microcontroller is used to control the operations for ECDSA, SHA-1, AES, PRNG, and the protocol handling according to ISO/IEC 7816-4. It shows that the entire implementation of AES (encryption and decryption) needs less than 2 400 GE of additional area. This design is smaller than existing (stand-alone) finite-state machine based implementations that are commonly integrated into the system over a common bus architecture.

Several investigations can be marked as future work. Performance improvements can be obtained by further optimizations especially on the algorithmic level of ECC implementations. New formulæ for ECC can reduce the number of multiplications and squarings, respectively, which has a significant impact in the total runtime of a scalar multiplication. Furthermore, binary-field elliptic curves might provide better performance results than prime-field curves. Formulæ for binary-field operations need only half of the multiplications compared to to their prime-field opponent. Note that the squaring operation over binary fields can be realized very efficient which also emphasizes the use of binary-field arithmetics.

RFID is one primer for the Internet of Things where many smart devices are connected to each other and the Internet. They form a common network to provide a more convenient life to users through transparent information transfer. However, this rises several security and privacy issues. So far, there do not exist standardized protocols for such services but there exist many proposals from the cryptographic community. A low-resource public-key processor presented in this thesis can thus provides a basis to realize narrow-strong and forward-privacy secure protocols [Vau07].

Bibliography

[AARR03] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM Side-channel(s). In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer, 2003.

[ABHW04] Harald Aigner, Holger Bock, Markus Hütter, and Johannes Wolkerstorfer. A Low-Cost ECC Coprocessor for Smartcards. In Marc Joye and Jean-Jacques Quisquater, editors, *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 107–118. Springer, 2004.

[AK96] Ross J. Anderson and Markus G. Kuhn. Tamper Resistance - a Cautionary Note. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce, Oakland, California, November 18-21, 1996*, pages 1–11. USENIX Association, November 1996. ISBN 1-880446-83-9.

[AK97] Ross J. Anderson and Markus G. Kuhn. Low Cost Attacks on Tamper Resistant Devices. In Bruce Christianson, Bruno Crispo, Mark Lomas, and Michael Roe, editors, *Security Protocols, 5th International Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 1997.

[Ame05] American National Standards Institute (ANSI). AMERICAN NATIONAL STANDARD X9.62-2005. Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), 2005.

[ARM97] ARM Ltd. AMBA Advanced Microcontroller Bus Architecture Specification. Available online at http://www.arm.com, 1997.

[AT03] Toru Akishita and Tsuyoshi Takagi. Zero-Value Point Attacks on Elliptic Curve Cryptosystem. In Colin Boyd and Wenbo Mao, editors, *Information Security, 6th International Conference, ISC 2003, Bristol, UK, October 1-3, 2003, Proceedings*, volume 2851

of *Lecture Notes in Computer Science*, pages 218–233. Springer, 2003.

[Atm07] Atmel Corporation. 8-bit AVR Microcontroller with 128K Bytes In-System Programmable Flash. Available online at http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf, August 2007.

[Aue08] Andreas Auer. Scaling Hardware for Electronic Signatures to a Minimum. Master thesis, University of Technology Graz, October 2008.

[BBD+08] Holger Bock, Michael Braun, Markus Dichtl, Erwin Hess, Johann Heyszl, Walter Kargl, Helmut Koroschetz, Bernd Meyer, and Hermann Seuschek. A Milestone Towards RFID Products Offering Asymmetric Authentication Based on Elliptic Curve Cryptography. Invited talk at RFIDsec 2008, July 2008.

[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 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceedings*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997.

[BGK+06] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede. Public-Key Cryptography for RFID-Tags. In *Workshop on RFID Security 2006 (RFIDSec06), July 12-14, Graz, Austria*, 2006.

[BGS+05] Steve Bono, Matthew Green, Adam Stubblefield, Ari Juels, Avi Rubin, and Michael Szydlo. Security Analysis of a Cryptographically-Enabled RFID Device. In *USENIX Security Symposium, Baltimore, Maryland, USA, July-August, 2005, Proceedings*, pages 1–16. USENIX, 2005.

[BHHM01] Michael K. Brown, Darrel R. Hankerson, Julio C. López Hernández, and Alfred J. Menezes. Software Implementation

of the NIST Elliptic Curves Over Prime Fields. In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001, The Cryptographers' Track at the RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings*, volume 2020 of *Lecture Notes in Computer Science*, pages 250–265. Springer, 2001.

[BJ02] Eric Brier and Marc Joye. Weierstraß Elliptic Curves and Side-Channel Attacks. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*, volume 2274 of *Lecture Notes in Computer Science*, pages 335–345. Springer, 2002.

[BJ03] Eric Brier and Marc Joye. Fast Point Multiplication on Elliptic Curves through Isogenies. In Marc Fossorier, Tom Høholdt, and Alain Poli, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes - AAECC 2003, 15th International Symposium, Toulouse, France, May 1216, 2003 Proceedings*, volume 2643 of *Lecture Notes in Computer Science*, pages 43–50. Springer, 2003.

[BL] Daniel Bernstein and Tanja Lange. Explicit-formulas database. www.hyperelliptic.org/EFD.

[Bla83] George Robert Blakely. A computer algorithm for calculating the product ab modulo m. *IEEE Transactions on Computers*, 32(5):497 – 500, 1983.

[BMM00] Ingrid Biehl, Bernd Meyer, and Volker Müller. Differential Fault Attacks on Elliptic Curve Cryptosystems. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 131–146. Springer, 2000.

[BMS+06] Lejla Batina, Nele Mentens, Kazuo Sakiyama, Bart Preneel, and Ingrid Verbauwhede. Low-Cost Elliptic Curve Cryptography for Wireless Sensor Networks. In Levente Buttyán, Virgil Gligor, and Dirk Westhoff, editors, *Security and Privacy in Ad-Hoc and Sensor Networks – ESAS 2006, Third European Workshop, Hamburg, Germany, September 20-21, 2006, Revised Selected Papers*, volume 4357, pages 6–17, Berlin Heidelberg, 2006. Springer-Verlag.

[BS97] Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.

[BS03] Johannes Blömer and Jean-Pierre Seifert. Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). In Rebecca N. Wright, editor, *Financial Cryptography, 7th International Conference, FC 2003, Guadeloupe, French West Indies, January 27-30, 2003, Revised Papers*, volume 2742 of *Lecture Notes in Computer Science*, pages 162–181. Springer, January 2003.

[Cad] Cadence Design Systems. The Cadence Design Systems Website. http://www.cadence.com/.

[Cer00a] Certicom Research. Standards for Efficient Cryptography, SEC 2: Recommended Elliptic Curve Domain Parameters, Version 1.0. Available online at http://www.secg.org/, September 2000.

[Cer00b] Certicom Research. Standards for Efficient Cryptography (SECG), SEC 1: Elliptic Curve Cryptography, Version 1.0. Available online at http://www.secg.org/, September 2000.

[CJ05] Mathieu Ciet and Marc Joye. Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults. *Des. Codes Cryptography*, 36(1):33–43, 2005. Available online at http://eprint.iacr.org/2003/028.pdf.

[CLP05] Dario Carluccio, Kerstin Lemke, and Christof Paar. Electromagnetic Side Channel Analysis of a Contactless Smart Card: First Results. In Elisabeth Oswald, editor, *Workshop on RFID and Lightweight Crypto (RFIDSec05), July 13-15, Graz, Austria*, pages 44–51, 2005.

[Cor99] Jean-Sébastien Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99, First International Workshop, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999.

[Cou09] Nicolas T. Courtois. The Dark Side of Security by Obscurity and Cloning MiFare Classic Rail and Building Passes Anywhere, Anytime. Cryptology ePrint Archive (http://eprint.iacr.org/), Report 2009/137, 2009.

[CRR03] 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 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2003.

[dKGHG08] Gerhard de Koning Gans, Jaap-Henk Hoepman, and Flavio D. Garcia. A Practical Attack on the MIFARE Classic. In G. Grimaud and F.-X. Standaert, editors, *Proceedings of the Eight Smart Card Research and Advanced Application Conference, CARDIS '08, September 8-11, 2008, London, UK, Proceedings*, volume 5189 of *Lecture Notes in Computer Science*, pages 267–282. Springer, September 2008.

[EGSG03] Hans Eberle, Nils Gura, Sheueling Chang Shantz, and Vipul Gupta. A Cryptographic Processor for Arbitrary Elliptic Curves over GF(2^m). In Ed Deprettere, Shuvra Bhattacharyya, Joseph Cavallaro, Alain Darte, and Lothar Thiele, editors, *Application-Specific Systems, Architectures, and Processors – ASAP 2003, IEEE 14th International Conference on Application-specific Systems, Architectures and Processors*, pages 444–454, The Hague, The Netherlands, June 2003.

[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 David Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008, Proceedings*, number 5157 in Lecture Notes in Computer Science, pages 203–220. Springer, 2008. ISBN 978-3-540-85173-8.

[EL09] Nevine Ebeid and Rob Lambert. Securing the Elliptic Curve Montgomery Ladder Against Fault Attacks. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2009, Lausanne, Switzerland, 2009, Proceedings*, pages 46–50, September 2009.

[ElG84] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology - CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.

[FAH+10] Martin Feldhofer, Manfred Josef Aigner, Michael Hutter, Thomas Plos, Erich Wenger, and Thomas Baier. Semi-Passive RFID Development Platform for Implementing and Attacking Security Tags. In *Workshop on RFID / USN Security and Cryptography - RISC 2010, November 9-10, London, UK, 2010.*, 2010.

[FDW04] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong Authentication for RFID Systems using the AES Algorithm. In Marc Joye and Jean-Jacques Quisquater, editors,

Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370. Springer, August 2004.

[Fel04] Martin Feldhofer. An Authentication Protocol in a Security Layer for RFID Smart Tags. In *12th IEEE Mediterranean Electrotechnical Conference (MELECON 2004), 12-15 May 2004, Dubrovnik, Croatia*, volume 2, pages 759–762. IEEE, May 2004.

[FGKS02] Wieland Fischer, Christophe Giraud, Erik Woodward Knudsen, and Jean-Pierre Seifert. Parallel scalar multiplication on general elliptic curves over Fp hedged against Non-Differential Side-Channel Attacks. Cryptology ePrint Archive (http://eprint.iacr.org/), Report 2002/007, 2002.

[FGM+10] Junfeng Fan, Xu Guo, Elke De Mulder, Patrick Schaumont, Bart Preneel, and Ingrid Verbauwhede. State-of-the-Art of Secure ECC Implementations: A Survey on known Side-Channel Attacks and Countermeasures. In *Hardware-Oriented Security and Trust - HOST 2010, In 3rd IEEE International Symposium, California, USA, June 13-14, 2010, Proceedings.*, pages 76–87. IEEE, 2010.

[FLRV08] Pierre-Alain Fouque, Reynald Lercier, Denis Réal, and Frédéric Valette. Fault Attack onElliptic Curve Montgomery Ladder Implementation. In Luca Breveglieri, Shay Gueron, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography, Fifth International Workshop, FDTC 2008, Washington DC, USA, August 10, 2008, Proceedings*, pages 92–98. IEEE Computer Society, August 2008.

[FP08] Martin Feldhofer and Thomas Popp. Power Analysis Resistant AES Implementation for Passive RFID Tags. In Christiopher Lackner, Timm Ostermann, Michael Sams, and Ronal Spilka, editors, *Proceedings of Austrochip 2008, October 8, 2008, Linz, Austria*, pages 1–6, October 2008. ISBN 978-3-200-01330-8.

[FS87] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - Crypto '86, 6th Annual International Cryptology Conference, Santa Barbara, California, USA, 1986, Proceeding*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Lecture Notes in Computer Science, Springer, 1987.

[FW07] Franz Fürbass and Johannes Wolkerstorfer. ECC Processor with Low Die Size for RFID Applications. In *Proceedings of 2007 IEEE*

International Symposium on Circuits and Systems. IEEE, IEEE, May 2007.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tyuyls, and Bart Preneel. Mutual Information Analysis - A Generic Side-Channel Distinguisher. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008, 10th International Workshop, Washington DC, USA, August 10-13, 2008, Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, August 2008.

[GdKGM+08] Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. Dismantling MIFARE Classic. In Sushil Jajodia and Javier Lopez, editors, *13th European Symposium on Research in Computer Security (ESORICS 2008), Malaga, Spain, 6-8 October, 2008, Proceedings (to appear)*, volume 5283 of *Lecture Notes in Computer Science*, pages 97–114. Springer Verlag, 2008.

[GES02] Nils Gura, Hans Eberle, and Sheueling Chang Shantz. Generic Implementations of Elliptic Curve Cryptography using Partial Reduction. In Vijay Atluri, editor, *Conference on Computer and Communications Security – CCS 2002, 9th ACM conference on Computer and Communications Security, Washington, DC, USA, November 17-21, 2002, Proceedings*, pages 108–116, Washington, DC, USA, November 2002. ACM Press.

[GFSV09] Xu Guo, Junfeng Fan, Patrick Schaumont, and Ingrid Verbauwhede. Programmable and Parallel ECC Coprocessor Architecture: Tradeoffs between Area, Speed and Security. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009, 11th International Workshop Lausanne, Switzerland, September 6-9, 2009 Proceedings*, volume 5747, pages 289–303. Springer Berlin / Heidelberg, September 2009.

[GG03] Catherine H. Gebotys and Robert J. Gebotys. Secure Elliptic Curve Implementations: An Analysis of Resistance to Power-Attacks in a DSP Processor. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 114–128. Springer, 2003.

[GHT05] Catherine H. Gebotys, Simon Ho, and Chin C. Tiu. EM Analysis of Rijndael and ECC on a Wireless Java-Based PDA. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware*

and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings, volume 3659 of *Lecture Notes in Computer Science*, pages 250–264. Springer, 2005.

[GJM10] Raveen Goundar, Marc Joye, and Atsuko Miyaji. Co-Z Addition Formulae and Binary Ladders on Elliptic Curves. In Stefan Mangard and Franois-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010, 12th International Workshop Santa Barbara, California, USA, August 1720, 2010 Proceedings*. Springer, 2010.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[Gou03] Louis Goubin. A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems. In Yvo Desmedt, editor, *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, volume 2567 of *Lecture Notes in Computer Science*, pages 199–210. Springer, 2003.

[GP08] Tim Güneysu and Christoph Paar. Ultra High Performance ECC over NIST Primes on Commercial FPGAs. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154, pages 62–78, Berlin, Heidelberg, 2008. Springer-Verlag.

[GPS06] Marc Girault, Guillaume Poupard, and Jacques Stern. On the fly authentication and signature schemes based on groups of unknown order. *Journal of Cryptology*, 19:463–487, 2006.

[GV10] Christophe Giraud and Vincent Verneuil. Atomicity Improvement for Elliptic Curve Scalar Multiplication. In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *Smart Card Research and Advanced Application - CARDIS 2010, 9th International Conference, Passau, Germany, April 14-16, 2010. Proceedings 2010*, volume 6035 of *Lecture Notes in Computer Science*, pages 80–101. Springer, 2010.

[GvRVS09] Flavio D. Garcia, Peter van Rossum, Roel Verdult, and Ronny Wichers Schreur. Wirelessly Pickpocketing a Mifare Classic Card. In *30th IEEE Symposium on Security and Privacy (S&P*

2009), Oakland, California, USA, 17-20 May, 2009, Proceedings, pages 3–15. IEEE Computer Society, May 2009.

[HAHH06] Panu Hämäläinen, Timo Alho, Marko Hännikäinen, and Timo D. Hämäläinen. Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core. In *9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools (DSD 2006), Dubrovnik, Croatia, 30. August-1 September, 2006. Proceedings*, pages 577–583. IEEE Computer Society, September 2006.

[Han08] Gerhard Hancke. Eavesdropping Attacks on High-Frequency RFID Tokens. In *Workshop on RFID Security 2008 (RFIDSec08), July 9-11, Budapest, Hungary*, volume RFIDsec 2008, pages 100–113, July 2008.

[HFP10] Michael Hutter, Martin Feldhofer, and Thomas Plos. An ECDSA Processor for RFID Authentication. In Berna Ors, editor, *Workshop on RFID Security - RFIDsec 2010, 6th Workshop, Istanbul, Turkey, June 7-9, 2010, Proceedings*, Lecture Notes in Computer Science. Springer, 2010.

[HFW11] Michael Hutter, Martin Feldhofer, and Johannes Wolkerstorfer. A Cryptographic Processor for Low-Resource Devices: Canning ECDSA and AES like Sardines. In *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems, Fifth International Workshop, WISTP 2011, Heraklion, Crete, Greece, June 1-3, 2011, Proceedings.*, 2011.

[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 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 320–333. Springer, September 2007.

[HMHW09] Michael Hutter, Marcel Medwed, Daniel Hein, and Johannes Wolkerstorfer. Attacking ECDSA-Enabled RFID Devices. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security - ACNS 2009, 7th International Conference, Paris-Rocquencourt, France, June 2-5, 2009, Proceedings*, volume 5536, pages 519–534. Springer, May 2009.

[HMP06] Engelbert Hubbers, Wojciech Mostowski, and Erik Poll. Tearing Java Cards. In *Proceedings of the 7th Edition of e-smart conference and demos, September 20-22, 2006 - Sophia Antipolis, French Riviera, France*, 2006.

[HMV04] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2004.

[HPF10] Michael Hutter, Thomas Plos, and Martin Feldhofer. On the Security of RFID Devices Against Implementation Attacks. *International Journal of Security and Networks 2010*, 5(2/3):106–118, 2010.

[HQ00] Gaël Hachez and Jean-Jacques Quisquater. Montgomery Exponentiation with no Final Subtractions: Improved Results. In Çetin Kaya Koç and Christoph Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop Worcester, MA, USA, August 1718, 2000 Proceedings*, volume 1965, pages 91–100. Springer Berlin / Heidelberg, August 2000.

[HSP08] Michael Hutter, Jörn-Marc Schmidt, and Thomas Plos. RFID and its Vulnerability to Faults. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008, 10th International Workshop, Washington DC, USA, August 10-13, 2008, Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 363–379. Springer, August 2008.

[HSP09] Michael Hutter, Jörn-Marc Schmidt, and Thomas Plos. Contact-Based Fault Injections and Power Analysis on RFID Tags. In *European Conference on Circuit Theory and Design 2009, ECCTD*, 2009.

[Hut09] Michael Hutter. RFID Authentication Protocols based on Elliptic Curves: A Top-Down Evaluation Survey. In Eduardo Fernández-Medina, Manu Malek, and Javier Hernando, editors, *International Conference on Security and Cryptography - SECRYPT 2009, it is part of ICETE - The International Joint Conference on e-Business and Telecommunications, Milan, Italy, July 7-10, 2009, Proceedings*, pages 101–110. INSTICC Press, 2009.

[HWF08a] Daniel Hein, Johannes Wolkerstorfer, , and Norbert Felber. ECC is Ready for RFID - A Proof in Silicon. In *Workshop on RFID Security 2008 (RFIDsec08)*, July 2008.

[HWF08b] Daniel Hein, Johannes Wolkerstorfer, and Norbert Felber. ECC is Ready for RFID - A Proof in Silicon. In *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, Canada, August 14-15, 2008, Revised Selected Papers*, Lecture Notes in Computer Science (LNCS), September 2008.

[IEE00] IEEE. IEEE Standard 1363-2000: IEEE Standard Specifications for Public-Key Cryptography. Available online at http://ieeexplore.ieee.org/servlet/opac?punumber=7168, 2000.

[IEE04] IEEE. IEEE Standard 1363a-2004: IEEE Standard Specifications for Public-Key Cryptography, Amendment 1: Additional Techniques. Available online at http://ieeexplore.ieee.org/servlet/opac?punumber=9276, September 2004.

[IMT02] Tetsuya Izu, Bodo Möller, and Tsuyoshi Takagi. Improved Elliptic Curve Multiplication Methods Resistant against Side Channel Attacks. In Alfred Menezes and Palash Sarkar, editors, *INDOCRYPT*, volume 2551 of *Lecture Notes in Computer Science*, pages 296–313. Springer, 2002.

[Int89] International Organisation for Standardization (ISO). ISO/IEC 7816: Identification cards - Integrated circuit(s) cards with contacts, 1989.

[Int93] International Organisation for Standardization (ISO). Information Technology - Security Techniques - Entity authentication mechanisms - Part 3: Entity authentication using a public key algorithm, 1993.

[Int95] International Organisation for Standardization (ISO). ISO/IEC 7816-4: Information technology - Identification cards - Integrated circuit(s) cards with contacts - Part 4: Interindustry commands for interchange. Available online at http://www.iso.org, 1995.

[Int99] International Organisation for Standardization (ISO). ISO/IEC 9798-2: Information technology – Security techniques – Entity authentication – Mechanisms using symmetric encipherment algorithms, 1999.

[Int01a] International Organisation for Standardization (ISO). ISO/IEC 10373-6: Identification cards - Test methods – Part 6: Proximity cards, 2001.

[Int01b] International Organisation for Standardization (ISO). ISO/IEC 15693-3: Identification cards - Contactless integrated circuit(s) cards - Vicinity cards – Part 3: Anticollision and transmission protocol, 2001.

[Int03] International Organisation for Standardization (ISO). ISO/IEC 7810: Identification cards – Physical characteristics, 2003.

[Int04a] International Organisation for Standardization (ISO). ISO/IEC 9798 Part 5: Information technology – Security techniques – Entity authentication – Mechanisms using zero knowledge techniques, December 2004.

[Int04b] International Organization for Standardization (ISO). ISO/IEC 18000-3: Information Technology AIDC Techniques — RFID for Item Management – Part 3: Parameters for air interface communications at 13.56 MHz, March 2004.

[Int06] International Organisation for Standardization (ISO). ISO/IEC 14888-3: Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms, 2006.

[Int08a] International Organization for Standardization (ISO). ISO/IEC 14443-4: Identification Cards - Contactless Integrated Circuit(s) Cards - Proximity Cards - Part4: Transmission Protocol. Available online at http://www.iso.org, 2008.

[Int08b] International Organization for Standardization (ISO). ISO/IEC 9594-8:2008: Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks, 2008.

[IT02] Tetsuya Izu and Tsuyoshi Takagi. A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks. In *5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002 Paris, France, February 1214, 2002 Proceedings*, volume 2274, pages 280–296. Springer-Verlag, 2002.

[JY03] Marc Joye and Sung-Ming Yen. The Montgomery Powering Ladder. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 291–302. Springer, 2003.

[Kae08] Hubert Kaeslin. *Digital Integrated Circuit Design – From VLSI Architectures to CMOS Fabrication*. Cambridge University Press, 2008. ISBN 978-0-521-88267-5.

[KAJ96] Çetin Kaya Koç, Tolga Acar, and Burton S. Kaliski Jr. Analyzing and Comparing Montgomery Multiplication Algorithms. *IEEE Micro*, 16(3):26–33, June 1996.

[Kal95] Burton Kaliski. The Montgomery Inverse and its Applications. *IEEE Transactions on Computers*, 44(8):1064–1065, August 1995.

[KF10] Thomas Kern and Martin Feldhofer. Low-Resource ECDSA Implementation for Passive RFID Tags. In *17th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2010), December 12-15th, 2010, Athens, Greece, Proceedings*, volume ?, pages ???–??? IEEE, 2010.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KK99] Oliver Kömmerling and Markus G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. In *Proceedings of the 1st USENIX Workshop on Smartcard Technology (Smartcard '99), Chicago, Illinois, USA, May 1011, 1999*, pages 9–20, McCormick Place South, May 1999. USENIX Association. ISBN 1-880446-34-0.

[KLW+06] Manuel Koschuch, Joachim Lechner, Andreas Weitzer, Johann Großschädl, Alexander Szekely, Stefan Tillich, and Johannes Wolkerstorfer. Hardware/Software Co-design of Elliptic Curve Cryptography on an 8051 Microcontroller. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 430–444. Springer, 2006.

[Koç95] Çetin Kaya Koç. RSA Hardware Implementation. Technical report, RSA Laboratories, RSA Data Security, Inc. 100 Marine Parkway, Suite 500 Redwood City, CA 94065-1031, 1995.

[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, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, number 1109 in Lecture Notes in Computer Science, pages 104–113. Springer, 1996.

[KOP09] Timo Kasper, David Oswald, and Christoph Paar. EM Side-Channel Attacks on Commercial Contactless Smartcards Using Low-Cost Equipment. In Heung Youl Youm and Moti Yung, editors, *Information Security Applications - WISA 2009, 10th International Workshop, Busan, Korea, August 25-27, 2009, Revised Selected Papers*, volume 5932 of *Lecture Notes in Computer Science*, pages 79–93. Springer, 2009.

[KP06] Sandeep S. Kumar and Christof Paar. Are standards compliant Elliptic Curve Cryptosystems feasible on RFID? In *Workshop on RFID Security 2006 (RFIDSec06), July 12-14, Graz, Austria*, 2006.

[KRCJ06] Mooseop Kim, Jaecheol Ryou, Yongje Choi, and Sungik Jun. Low Power AES Hardware Architecture for Radio Frequency Identification . In Hiroshi Yoshiura, Kouichi Sakurai, Kai Rannenberg, Yuko Murayama, and Shinichi Kawamura, editors, *First International Workshop on Security (IWSEC 2006), Kyoto, Japan, October 23-24, 2006, Proceedings*, volume 4266 of *Lecture Notes in Computer Science*, pages 353–363. Springer, October 2006.

[KS06] Jens-Peter Kaps and Berk Sunar. Energy comparison of AES and SHA-1 for ubiquitous computing. In Xiaobo Zhou, Oleg Sokolsky, LuYan, Eun-Sun Jung, Zili Shao, Yi Mu, Dong-Chun Lee, Daeyoung Kim Young-Sik Jeong, and Cheng-Zhong Xu, editors, *2nd IFIP International Symposium on Network Centric Ubiquitous Systems (NCUS 2006), Seoul, Korea, August 1-4, 2006, Proceedings*, volume 4097 of *Lecture Notes in Computer Science*, pages 372–381. Springer, 2006.

[LH04] Chae Hoon Lim and Hyo Sun Hwang. Fast Implementation of Elliptic Curve Arithmetic in GF(pn). In *Third International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2000, Melbourne, Victoria, Australia, January 18-20, 2000. Proceedings*, volume 1751, pages 405–421. Springer Berlin / Heidelberg, 2004.

[LSBV08] Yong Ki Lee, Kazuo Sakiyama, Lejla Batina, and Ingrid Verbauwhede. Elliptic-Curve-Based Security Processor for RFID. *IEEE Transactions on Computers*, 57(11):1514–1527, November 2008.

[Man03] Stefan Mangard. Exploiting Radiated Emissions - EM Attacks on Cryptographic ICs. In Timm Ostermann and Christoph Lackner, editors, *Proceedings of Austrochip 2003, October 3, 2003, Linz, Austria*, pages 13–16, October 2003. ISBN 3-200-00021-X.

[Med07] Marcel Medwed. Template-Based SPA Attacks on 32-bit ECDSA Implementations. Master's thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, December 2007.

[Mel06] Nicolas Meloni. Fast and Secure Elliptic Curve Scalar Multiplication Over Prime Fields Using Special Addition Chains. Cryptology ePrint Archive, Report 2006/216, 2006.

[MMM04] Ciaran McIvor, Maire McLoone, and John McCanny. An FPGA Elliptic Curve Cryptographic Accelerator over GF(p). In *Irish Signals and Systems Conference – ISSC 2004, Belfast, Ireland, 30 June-2 July 2004*, pages 589–594. IET, June 2004.

[Mon85] Peter L. Montgomery. Modular Multiplication without Trial Division. *Mathematics of Computation*, 44:519–521, 1985.

[Mon87] Peter L. Montgomery. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, 48(177):243–264, January 1987. ISSN 0025-5718.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks – Revealing the Secrets of Smart Cards*. Springer, 2007. ISBN 978-0-387-30857-9.

[MvOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. Series on Discrete Mathematics and its Applications. CRC Press, 1997. ISBN 0-8493-8523-7, Available online at http://www.cacr.math.uwaterloo.ca/hac/.

[Nat99] National Institute of Standards and Technology (NIST). FIPS-46-3: Data Encryption Standard, October 1999. Available online at http://www.itl.nist.gov/fipspubs/.

[Nat00] National Institute of Standards and Technology (NIST). FIPS-186-2: Digital Signature Standard (DSS), January 2000. Available online at http://www.itl.nist.gov/fipspubs/.

[Nat01] National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard, November 2001. Available online at http://www.itl.nist.gov/fipspubs/.

[Nat02] National Institute of Standards and Technology (NIST). FIPS-180-2: Secure Hash Standard, August 2002. Available online at http://www.itl.nist.gov/fipspubs/.

[Nat08] National Institute of Standards and Technology (NIST). FIPS-180-3: Secure Hash Standard, October 2008. Available online at http://www.itl.nist.gov/fipspubs/.

[Nat09] National Institute of Standards and Technology (NIST). FIPS-186-3: Digital Signature Standard (DSS), 2009. Available online at http://www.itl.nist.gov/fipspubs/.

[NESP08] Karsten Nohl, David Evans, Starbug, and Henryk Plötz. Reverse-Engineering a Cryptographic RFID Tag. In *USENIX Security Symposium, San Jose, CA, USA, 31 July, 2008, Proceedings*, pages 1–9. USENIX, 2008.

[NH09] Christop Nagl and Michael Hutter. Coupon Recalculation for the Schnorr and GPS Identification Scheme: A Performance Evaluation. In *Workshop on RFID Security 2009 - RFIDSec 2009, 5th edition, Leuven, Belgium, June 30-July 2, 2009*, 2009.

[Noh08] Karsten Nohl. Cryptanalysis of Crypto-1. Computer Science Department University of Virginia, White Paper, 2008.

[OBP02] Siddika Berna Örs, Lejla Batina, and Bart Preneel. Hardware Implementation of Elliptic Curve Processor over GF(p). *International Journal of Embedded Systems*, 3(4):229–240, 2002.

[Oka93] Tatsuaki Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer, 1993.

[OOP03] Siddika Berna Örs, Elisabeth Oswald, and Bart Preneel. Power-Analysis Attacks on FPGAs – First Experimental Results. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2003.

[OP00] Gerardo Orlando and Christoph Paar. A High-Performance Reconfigurable Elliptic Curve Processor for GF(2^m). In Çetin Kaya Koç and Christoph Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop Worcester, MA, USA, August 1718, 2000 Proceedings*, volume 1965 of *0302-9743*, pages 41–56, London, UK, August 2000. Springer Berlin / Heidelberg.

[OP01] Gerardo Orlando and Christoph Paar. A Scalable GF(p) Elliptic Curve Processor Architecture for Programmable Hardware. In Çetin Kaya Koç, David Naccache, and Christoph Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001, Third International Workshop Paris, France, May 1416, 2001 Proceedings*, volume 2162, pages 348 –363, London, UK, August 2001. Springer Berlin / Heidelberg.

[OS00] Katsuyuki Okeya and Kouichi Sakurai. Power Analysis Breaks Elliptic Curve Cryptosystems Even Secure against the Timing Attack. In Bimal K. Roy and Eiji Okamoto, editors, *Progress in Cryptology - INDOCRYPT 2000, First International Conference in Cryptology in India, Calcutta, India, December 10-13, 2000, Proceedings.*, volume 1977 of *Lecture Notes In Computer Science*, pages 178–190. Springer, 2000.

[OS06] Yossef Oren and Adi Shamir. Remote Power Analysis of RFID Tags. Master's thesis, Weizmann Institute of Science, Rehovot,

Israel, August 2006. Further information on http://www.wisdom.weizmann.ac.il/~yossio/rfid/.

[OS07] Yossef Oren and Adi Shamir. Remote Password Extraction from RFID Tags. *IEEE Transactions on Computers*, 56(9):1292–1296, September 2007.

[ÖSS04] Erdinc Öztürk, Berk Sunar, and Erkay Savas. Low-Power Elliptic Curve Cryptography Using Scaled Modular Arithmetic. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 92–106. Springer, August 2004.

[OTIT00] Souichi Okada, Naoya Torii, Kouichi Itoh, and Masahiko Takenaka. Implementation of Elliptic Curve Cryptographic Coprocessor over GF(2^m) on an FPGA. In Çetin Kaya Koç and Christoph Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop Worcester, MA, USA, August 1718, 2000 Proceedings*, volume 1965, pages 25–40, London, UK, August 2000. Springer Berlin / Heidelberg.

[Ott05] Martin Otto. *Fault Attacks and Countermeasures*. PhD thesis, Universität Paderborn, 2005.

[P+03] Bart Preneel et al. NESSIE Security Report, D20, 2003. Available online at http://www.nessie.eu.org.

[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 2008 (RFIDSec08), July 9-11, Budapest, Hungary*, pages 114–127, July 2008.

[PHF09] Thomas Plos, Michael Hutter, and Martin Feldhofer. On Comparing Side-Channel Preprocessing Techniques for Attacking RFID Devices. In Heung Youl Youm and Moti Yung, editors, *Information Security Applications, 10th International Workshop, WISA 2009, Busan, Korea, August 25-27, 2009, Revised Selected Papers*, volume 5932 of *Lecture Notes in Computer Science*, pages 163–177. Springer, December 2009.

[PHH08] Thomas Plos, Michael Hutter, and Christoph Herbst. Enhancing Side-Channel Analysis with Low-Cost Shielding Techniques. In Michael Sams Christoph Lackner, Timm Ostermann and Ronald Spilka, editors, *Proceedings of Austrochip 2008, October 8, 2008, Linz, Austria*, pages 90–95, October 2008. ISBN 978-3-200-01330-8.

[Plo08] Thomas Plos. Susceptibility of UHF RFID Tags to Electromagnetic Analysis. In Tal Malkin, editor, *Topics in Cryptology - CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, April 8-11, 2008, Proceedings*, volume 4964 of *Lecture Notes in Computer Science*, pages 288–300. Springer, April 2008.

[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 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 2005.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In Isabelle Attali and Thomas P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[QS02] Jean-Jacques Quisquater and David Samyde. Eddy Current for Magnetic Analysis with Active Sensor. In *Proceedings of the 3rd International Conference on Research in SmartCards (E-Smart'02), Nice, France, September, 2002*, pages 185–194. UCL, September 2002.

[Rob06] Harko Robroch. ePassport Privacy Attack. Riscure presentation at Cards Asia Singapore, April 2006., 2006.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978. ISSN 0001-0782.

[SA03] Sergei P. Skorobogatov and Ross J. Anderson. Optical Fault Induction Attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer, 2003.

[SBG+03] Richard Schroeppel, Cheryl Beaver, Rita Gonzales, Russell Miller, and Tim Draelos. A Low-Power Design for an Elliptic Curve Digital Signature Chip. In Burton S. Kaliski Jr.,

Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 366–380. Springer, 2003.

[Sch] Rhode & Schwarz. Website Rhode & Schwarz.

[Sch90] Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceeding*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1990.

[Sch09] Jörn-Marc Schmidt. *Implementation Attacks - Manipulating Devices to Reveal Their Secrets*. PhD thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, September 2009.

[SH07] Jörn-Marc Schmidt and Michael Hutter. Optical and EM Fault-Attacks on CRT-based RSA: Concrete Results. In Karl Christian Posch and Johannes Wolkerstorfer, editors, *Proceedings of Austrochip 2007, October 11, 2007, Graz, Austria*, pages 61–67. Verlag der Technischen Universität Graz, October 2007. ISBN 978-3-902465-87-0.

[SHP09] Jörn-Marc Schmidt, Michael Hutter, and Thomas Plos. Optical Fault Attacks on AES: A Threat in Violet. In David Naccache and Elisabeth Oswald, editors, *Fault Diagnosis and Tolerance in Cryptography, Sixth International Workshop, FDTC 2009, Lausanne, Switzerland September 6, 2009, Procceedings*, pages 13–22. IEEE-CS Press, September 2009.

[Sid] Side-channel attack standard evaluation board. The SASEBO Website. http://www.rcis.aist.go.jp/special/SASEBO/.

[Sko05] Sergei P. Skorobogatov. *Semi-invasive attacks - A new approach to hardware security analysis*. PhD thesis, University of Cambridge - Computer Laboratory, 2005. Available online at http://www.cl.cam.ac.uk/TechReports/.

[Slo85] Kenneth R. Sloan. Comments on "a computer algorithm for calculating the product ab modulo m". *IEEE Transactions on Computers*, 34:290 – 292, March 1985.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In

Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.

[Sol99] Jerome A. Solinas. Generalized mersenne numbers. Technical Report CORR 99-39, Centre for Applied Cryptographic Research, University of Waterloo, 1999.

[SS05] Yasuyuki Sakai and Kouichi Sakurai. Simple Power Analysis on Fast Modular Reduction with NIST Recommended Elliptic Curves. In *Information and Communications Security 7th International Conference - ICICS 2005, Beijing, China, December 10-13, 2005. Proceedings*, volume 3783, pages 169–180. Springer, 2005.

[ST03] Akashi Satoh and Kohji Takano. A Scalable Dual-Field Elliptic Curve Cryptographic Processor. *IEEE Transactions on Computers*, 52(4):449–460, 2003.

[TAV02] Kris Tiri, Moonmoon Akmal, and Ingrid Verbauwhede. A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. In *28th European Solid-State Circuits Conference - ESSCIRC 2002, Florence, Italy, September 24-26, 2002, Proceedings*, pages 403–406. IEEE, September 2002.

[The] The Mathworks. MATLAB - The Language of Technical Computing. http://www.mathworks.com/products/matlab/.

[TKL05] Elena Trichina, Tymur Korkishko, and Kyung-Hee Lee. Small Size, Low Power, Side Channel-Immune AES Coprocessor: Design and Synthesis Results. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *Advanced Encryption Standard - AES, 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers*, volume 3373 of *Lecture Notes in Computer Science*, pages 113–127. Springer, 2005.

[TTO97] K.T. Tan, S.H. Tan, and S.H. Ong. Functional failure analysis on analog device by optical beam induced current technique. In M.K. Radhakrishnan, Philip Ho, and Chim Wai Kin, editors, *Proceedings of the 6th International Symposium on Physical and Failure Analysis of Integrated Circuits (IPFA97), Raffles City Convention Centre, Singapore, July 21-25, 1997*, pages 296–301, Raffles City Convention Centre, Singapore, July 1997. IEEExplore, IEEE.

[uHK04] Thomas Finke und Harald Kelter. Abhörmöglichkeiten der Kommunikation zwischen Lesegerät und Transponder am Beispiel eines ISO14443-Systems. Whitepaper des BSI available under http://www.bsi.de/fachthem/rfid/Abh_RFID.pdf, German, 2004.

[Vau07] Serge Vaudenay. On Privacy Models for RFID. In *Conference on the Theory and Application of Cryptology and Information Security - ASIACRYPT 2007, 13th International Conference, Kuching, Malaysia, December 2-6, 2007. Proceedings*, volume 4833, pages 68–87, 2007.

[VMG+10] Jo Vliegen, Nele Mentens, Jan Genoe, An Braeken, Serge Kubera, Adbdellah Touhafi, and Ingrid Verbauwhede. A compact FPGA-based Architecture for Elliptic Curve Cryptography over Prime Fields. In *ASAP 2010 21st IEEE International Conference on Application-specific Systems, Architectures and Processors, Rennes, France, July 7-9, 2010*, 2010.

[Wal99] Colin D. Walter. Montgomery's Multiplication Technique: How to Make it Smaller and Faster. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES99, First International Workshop, Worcester, MA, USA, August 1213, 1999 Proceedings.*, volume 1717, pages 80–93. Springer, 1999.

[WFF11] Erich Wenger, Martin Feldhofer, and Norbert Felber. Low-Resource Hardware Design of an Elliptic Curve Processor for Contactless Devices. In Yongwha Chung and Moti Yung, editors, *Information Security Applications*, volume 6513, pages 92–106. Springer Berlin / Heidelberg, 2011.

[Wol03] Johannes Wolkerstorfer. Dual-Field Arithmetic Unit for GF(p) and GF(2^m). In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 500–514. Springer, 2003.

[Wol05] Johannes Wolkerstorfer. Is Elliptic-Curve Cryptography Suitable for Small Devices? In *Workshop on RFID and Lightweight Crypto, July 13-15, 2005, Graz, Austria*, pages 78–91, 2005.

[YJ00] Sung-Ming Yen and Marc Joye. Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis. In *IEEE Transactions on Computers*, volume 49 of *IEEE Transactions on Computers*, pages 967–970. IEEE Computer Society, 2000.

Index

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……..
date (signature)