Horizontal address-bit SCA attacks against ECC and appropriate countermeasures

Von der Fakultät 1 - MINT - Mathematik, Informatik, Physik,
Elektro- und Informationstechnik
der Brandenburgischen Technischen Universität Cottbus–Senftenberg
genehmigte Dissertation
zur Erlangung des akademischen Grades eines

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

vorgelegt von

Diplom-Ingenieur (Dipl.-Ing.)
Ievgen Kabin

Vorsitzende/r: Prof. Dr. Andriy Panchenko
Gutachter/in: Prof. Dr. Peter Langendörfer
Gutachter/in: Prof. Dr. Nicolas Sklavos
Gutachter/in: Prof. Dr. Miloš Krstić

Tag der mündlichen Prüfung: 12. Mai 2023

DOI: https://doi.org/10.26127/BTUOpen-6397

Acknowledgements

This work is partially developed under the funding of *fast sign* project (funded by the Federal Ministry of Education and Research of Germany under grant number 03ZZ0527A) and the internal IHP project *Totale Resilience*.

Abstract

There are two types of elliptic curves (ECs) standardized for use in cryptographic protocols: ECs over extended binary fields $GF(2^n)$ and ECs over prime fields $GF(p)$. The $k\boldsymbol{P}$ operation is the most time and energy-consuming operation in elliptic curve cryptography (ECC) protocols. For large networks or time-critical applications, fast execution and low-energy consumption are essential for cryptographic operations. An additional requirement for applications, which are sensitive to manipulations, is that the implemented algorithms have to be resistant to side-channel analysis (SCA) attacks if an attacker can have physical access to the working cryptographic device. Highly regular Montgomery $k\boldsymbol{P}$ algorithm as well as $k\boldsymbol{P}$ algorithms based on the atomicity principle are in the literature reported as resistant against simple SCA attacks.

In this work we investigated the resistance of different $k\boldsymbol{P}$ implementations based on the Montgomery ladder against horizontal, i.e. single trace, attacks. Applying statistical methods for the analysis we were able to reveal the secret value k completely. The reason causing the success of our attacks is the key-dependent addressing of the registers and other design blocks, which is an inherent feature of binary $k\boldsymbol{P}$ algorithms. This dependency was successfully exploited in the past by Itoh et al. analyzing many hundreds of $k\boldsymbol{P}$ traces, i.e. this attack is a vertical address-bit differential power analysis attack against Montgomery ladder. The vulnerability of the Montgomery ladder against horizontal address-bit attacks was detected and demonstrated during our investigations. We were able to reveal the scalar k exploiting the address-bit vulnerability in single trace attacks using not only statistical methods, but also Fourier transform, selected clustering methods as well as one of the simplest methods – the automatized simple SCA. We performed successful horizontal address-bit SCA attacks against both types of ECs, i.e. against highly regular Montgomery ladder and against a binary $k\boldsymbol{P}$ algorithm implementing atomic patterns. The success of our attacks shows that the regularity and atomicity principles are not effective against horizontal address-bit attacks. As a means for reducing the attack success, we investigated the hiding ability of the field multiplier which is usually the largest block of $k\boldsymbol{P}$ designs. We implemented our field multiplier for ECs over prime fields corresponding to the 4-segment Karatsuba multiplication formula that reduces the execution time and the energy consumption for a $k\boldsymbol{P}$ operation by about 40 % in comparison to multipliers exploiting the classical multiplication formula. However, the energy consumption per clock remained in our multiplier without significant changes, i.e. the protective hiding properties of the multiplier as a noise source were not decreased. Another advantage of our field multiplier is its inherent resistance to horizontal collision attacks, in contrast to multipliers based on the classic multiplication formula. To the best of our knowledge, it is the first implementation of a $k\boldsymbol{P}$ accelerator for ECs over $GF(p)$ exploiting the advantages of the iterative 4-segment Karatsuba multiplication method. Additionally, we proposed regular scheduling for the block addressing as an effective strategy for reducing the success of horizontal address-bit attacks. Combining this approach with the hiding features of the field multipliers can increase the resistance of the $k\boldsymbol{P}$ designs for both types of ECs against a broad spectrum of SCA attacks. The mentioned analysis methods can be successfully applied for determining SCA leakage sources in the early design phase. Based on the investigations within this work, some ideas for increasing the resistance of cryptographic implementations were registered as 2 patents. The horizontal attacks can be successfully applied to determine side-channel leakage sources early in the design phase. The protection mechanisms proposed here can be a basis for automating the design process. It can be helpful for engineering side-channel resistant cryptographic implementations.

Kurzfassung

Es gibt zwei Arten von elliptischen Kurven (ECs), die für die Verwendung in kryptografischen Protokollen standardisiert sind: ECs über erweiterten binären Galoiskörper $GF(2^n)$ und ECs über Primkörper $GF(p)$. Die skalar Multiplikation bekannt als $k\boldsymbol{P}$-Operation ist die zeit- und energieaufwändigste Operation in Protokollen der elliptischen Kurven Kryptographie. Bei großen Netzwerken oder zeitkritischen Anwendungen sind eine schnelle Ausführung und ein geringer Energieverbrauch für kryptografische Operationen unerlässlich. Eine weitere Anforderung an kryptographische Implementierungen ist ihre Resistenz gegen Seitenkanalangriffe, wenn ein Angreifer physischen Zugang zu funktionierenden Geräten haben kann. Hochreguläre Implementierungen nach dem Montgomery kP-Algorithmus sowie auch die Implementierungen nach dem *atomicity* Prinzip wurden in der Literatur als resistent gegen simple Seitenkanalangriffe beschrieben.

In dieser Arbeit haben wir die Resistenz verschiedener, auf der Montgomery-Ladder basierenden, $k\boldsymbol{P}$-Implementierungen gegen Single-Trace-Angriffe untersucht. Single-Trace-Angriffe sind auch als horizontale Angriffe bekannt. Mittels statistischer Analyse wurde der Wert des Skalars k in unseren Angriffen erfolgreich extrahiert. Die Ursache des Erfolges unserer Angriffe ist die schlüsselabhängige Adressierung der Register und Designblöcke, die ein inhärenter Teil der binären $k\boldsymbol{P}$-Algorithmen ist. Diese Abhängigkeit wurde in der Vergangenheit von Itoh et al. erfolgreich verwendet, um den geheimen Skalar k mittels differenzieller Analyse mehrerer Traces zu extrahieren. Dieser Angriff ist als vertikale Adress-Bit Differential Power Analyse gegen die Montgomery-Ladder bekannt. Die Vulnerabilität des Montgomery-Ladders im Hinblick auf horizontale Adress-Bit Seitenkanalangriffe wurde im Rahmen unserer Untersuchungen entdeckt und demonstriert. Wir haben für das Extrahieren des Skalars k nicht nur statistischen Analyse-Methoden, sondern auch die Fourier-Transformation, ausgewählte Clustering-Methoden sowie simple Seitenkanalangriffe, die wir automatisiert haben, erfolgreich verwendet. Die horizontalen Adress-Bit Angriffe wurden erfolgreich gegen beide Arten der elliptischen Kurven durchgeführt. Angegriffen wurde nicht nur die hochreguläre Montgomery Ladder für die ECs über $GF(2^n)$, sondern auch ein *atomic pattern* $k\boldsymbol{P}$-Algorithmus für die ECs über $GF(p)$. Der Erfolg der durchgeführten Angriffe demonstriert, dass die *regularity* und *atomicity* Prinzipien als alleinige Gegenmaßnahmen gegen horizontale Angriffe nicht wirksam sind. Als eine weitere Schutzmaßnahme wurde in dieser Arbeit die Aktivität des Feldmultiplizierers vorgeschlagen und untersucht. Der Feldmultiplizierer ist normalerweise der größte Block in $k\boldsymbol{P}$-Designs, benötigt viel Zeit und Energie für die Produktberechnung und kann somit als eine interne Rauschquelle genutzt werden, um die Aktivität anderer Designblöcke zu verbergen. Um die Ausführungszeit und den Energieverbrauch der gesamten $k\boldsymbol{P}$-Operation zu reduzieren, haben wir den Feldmultiplizierer für ECs über Primkörper nach der 4-Segment-Karatsuba-Multiplikationsformel implementiert. Das reduziert die Ausführungszeit und – dementsprechend auch – den Energieverbrauch für eine $k\boldsymbol{P}$-Operation um etwa 40 % im Vergleich zur klassischen Multiplikationsformel. Der Energieverbrauch pro Takt bleibt aber ohne wesentliche Änderungen, d.h. die Schutzeigenschaften des Multiplizierers als Rauschquelle wurden nicht beeinträchtigt. Ein weiterer Vorteil unseres Multiplizierers ist seine inhärente Resistenz gegen horizontale Korrelationsangriffe, im Gegensatz zu den Multiplieren auf Basis der klassischen Multiplikationsformel. Nach unserem besten Wissen ist der hier untersuchte $k\boldsymbol{P}$-Beschleuniger für ECs über $GF(p)$ der erste, der die Vorteile der iterativen 4-Segment Karatsuba Multiplikationsmethode nutzt. Außerdem wurde

hier eine regelmäßige Adressierung der Designblöcke als effektive Strategie zur Reduzierung des Erfolgs horizontaler Adress-Bit Angriffe vorgeschlagen, implementiert und untersucht. Die Kombination der regelmäßigen Adressierung mit den Hiding-Eigenschaften des Feldmultiplizierers kann die Resistenz der in Hardware implementierten $k\boldsymbol{P}$-Designs gegen ein breites Spektrum von Seitenkanalangriffen signifikant erhöhen. Die im Rahmen dieser Arbeit durchgeführten Untersuchungen zur Erhöhung der Resistenz kryptographischer Implementierungen bilden die Basis für 2 eingereichte Patente. Horizontale Angriffe können in frühen Designphasen erfolgreich zur Bestimmung von *side-channel leakage sources* verwendet werden. Die hier vorgeschlagenen Schutzmechanismen können eine Basis für die Automatisierung des Designprozesses bilden, und so helfen, seitenkanalresistente kryptographische Implementierungen zu entwickeln.

Glossary

AC	Alternating current
AES	Advanced encryption standard
ALU	Arithmetic logic unit
ASIC	Application-specific integrated circuit
CMOS	Complementary metal-oxide semiconductor
DC	Direct current
DES	Data encryption standard
DH	Diffie-Hellman protocol
DoM	Difference of means
DPA	Differential power analysis
EC	Elliptic curve
ECC	Elliptic curve cryptosystem
ECDH	Elliptic curve Diffie-Hellman
ECDSA	Elliptic curve digital signature algorithm
EM	Electromagnetic
EMA	Electromagnetic analysis
EMT	Electromagnetic trace
FFT	Fast Fourier transform
FPGA	Field-programmable gate array
GALS	Globally asynchronous locally synchronous
GC	Gate complexity
GF	Galois field
GS	Gigasamples
GUI	Graphical user interface
LUT	Lookup Table
MM	Multiplication method
NIST	National Institute of Standards and Technology
PCB	Printed circuit board
PT	Power trace
RSA	RSA (Rivest–Shamir–Adleman) cryptosystem
RTL	Register-transfer level
SCA	Side-channel Analysis
SEMA	Simple electromagnetic analysis
SPA	Simple power analysis
VHDL	Very high-speed integrated circuit hardware description language

Table of contents

1 Introduction

1.1 Motivation

In recent years the number of networked devices and the need for the machine to machine communication were increasing dramatically and the prognoses are that this trend will be aggravated by 5G. In order to ensure correct system behavior security features such as confidentiality, data integrity and authentication are essential. This holds not only true for networked devices but also for complex systems such as telemedicine appliances that are compiled of costly parts from highly renowned manufacturers.

Cryptographic approaches are usual means implementing a secure communication. If the *secret* key of a communication session is revealed the confidentiality of the communication is lost. Asymmetric cryptographic approaches [1] such as RSA and elliptic curve cryptography (ECC) are key when it comes to ensuring data integrity, authentication and non-repudiation. If the *private* key of an entity is revealed, the identity of this entity can be stolen. Due to this and also to the fact that the asymmetric cryptographic approaches are usually the basis for the exchange of symmetric keys, implementations of asymmetric cryptographic protocols have to be resistant against a bright spectrum of attacks.

In comparison to RSA, the length of cryptographic keys for ECC is significantly smaller. This results in the reduced execution time and energy consumption for cryptographic operations, for sending/receiving of the encrypted data as well as digital signatures as well as a reduced area of a hardware implementation of cryptographic accelerators. These advantages make ECC applicable for resource-constrained devices such as sensor nodes, Wireless Sensor Networks (WSN) and Internet of Things (IoT), especially if elliptic curve (EC) cryptographic protocols are accelerated, i.e. if complex operations are implemented in hardware.

For the currently standardized and used cryptographic algorithms it is common sense that they cannot be brute forced or broken by cryptanalysis in any reasonable amount of time. However, the situation changes drastically when the attacked devices are physically accessible. With the advent of the Internet of Things a lot of devices are no longer physically protected. Devices can be stolen, analysed in a well-equipped laboratory and returned. Measurements made on the devices or even manipulations might not be noticed by the owner of the attacked device. Physical attacks are relatively easy to prepare and execute, are usually much more effective than brute force attacks, and must be considered in the early stage when designing cryptographic implementations.

Side Channel Analysis (SCA) attacks are a kind of physical attacks. They exploit the fact that physical parameters such as time, current, electromagnetic radiation of cryptographic devices, etc. can be measured during the execution of cryptographic operations. The shape of measured power traces (PT) or electromagnetic traces (EMT) depends not only on the implemented circuit (manufacturing technology, applied gates and their placement, area of the design, etc.) but also on the processed input data and the used cryptographic key. Thus, the measured traces can be analysed with the goal to reveal the *secret*/*private* key. Sometimes the used cryptographic key can be successfully revealed analysing only a single measured trace. Such single-trace attacks are also known as horizontal attacks. Power analysis (PA) and electromagnetic analysis (EMA) attacks are most commonly applied nowadays.

To be applicable for a lot of tiny, resources constrained devices in a near future, asymmetric cryptographic implementations, especially hardware accelerators, have to be resistant against a broad spectrum of physical attacks, especially against horizontal power analysis and electromagnetic analysis attacks.

1.2 Structure of this thesis

This thesis is structured as follows. In section 2 we are focusing on the basics of elliptic curve cryptography and state-of-the-art attacks and countermeasures relevant for this work. We describe the hardware IHP accelerator for elliptic curve point multiplication $k\boldsymbol{P}$, which we used as the starting point of this thesis. Additionally, we explain shortly the methods and tools for the analysis of the design's execution traces, which were available at IHP at the beginning of this work. The details given in the section are necessary for understanding the ideas proposed and evaluated in this work.

In section 3 we explain all the details regarding the collection of the traces. In our investigation we used simulated power traces of $k\boldsymbol{P}$ executions as well as measured power and electromagnetic traces. We explain here detailed how we obtained the simulated traces and we discuss many aspects of measurements that can influence the success of SCA attacks. During this work we implemented, applied and improved many methods for the statistical analysis of the traces. The implemented methods are the basis of the IHP SCA tool. The implemented methods, as well as the criteria applied for evaluation of the attack success, are described detailed in this section.

In section 4 we discuss the vulnerability of the basic IHP $k\boldsymbol{P}$ design. We attacked functional blocks of the design and estimated the influence of the blocks on the success of the attack using the trace of the whole design. Based on these experiments, we made our assumption regarding the main source of side-channel leakage in the basic IHP $k\boldsymbol{P}$ design and defined a strategy to increase its resistance.

We evaluated the success of horizontal SCA attacks applied to the designs protected by the existing state-of-the-art countermeasures in section 5. We showed that well-known randomization techniques such as EC point blinding and key randomization are not effective against horizontal attacks. We evaluated also the resistance of a globally asynchronous locally synchronous $k\boldsymbol{P}$ design against simple power analysis (SPA) attack as well as attacked an open-source $k\boldsymbol{P}$ design implementing a unified formula for EC point doublings and point additions and randomization of the sequence of these operations in the Montgomery ladder. In all of these cases we were able to reveal the processed scalar completely.

Following our strategy, defined in section 4, we started to seek a noise source that is an inherent part of the design and not an external one. We decided to increase our design's resistance by maximizing the noise produced by the multiplier. Therefore, in section 6 we investigated the hiding effect of multipliers implemented using different multiplication formulae. In addition we proposed a regular scheduling for the addressing of design blocks to reduce the influence of the main leakage source. Using simulated power traces we evaluated the proposed approaches as well as their combination. Our ideas were evaluated in section 7 using measured power and/or electromagnetic traces. We performed measurements of the $k\boldsymbol{P}$ traces on an FPGA as well as on application-specific integrated circuits (ASICs) manufactured in the IHP 250 nm technology.

Thus, in sections 3-7 we investigated the vulnerability of the basic IHP $k\boldsymbol{P}$ design to horizontal attacks, the sources and the nature of the SCA leakage. This design is an implementation of the Montgomery ladder for EC over extended binary finite fields that is a highly regular algorithm. We show, that the reason of the vulnerability is the key-dependent addressing of design blocks, and that it can be successfully exploited in horizontal, i.e. single-trace, attacks. Due to the nature of the vulnerability, we assumed that not only the Montgomery ladder but also algorithms implementing atomicity principles can be vulnerable to horizontal attacks as well. We also assumed that the methods increasing the resistance of the Montgomery ladder can be successfully applied by the implementation of $k\boldsymbol{P}$ algorithms with atomic patterns. In section 8 we discuss the vulnerability of the $k\boldsymbol{P}$ designs for ECs over prime finite fields that we implemented corresponding to the atomic patterns described in [2]. We show that our

strategy, i.e. using the hiding capability of the field multiplier, can reduce the success of horizontal attacks significantly.

Finally, the work ends with conclusions, where we summarize our main results and share our plans for future work.

1.3 List of publications

This work is based mostly on the results obtained during my work at IHP in the project *fast sign* and in the internal IHP project *Totale Resilience*. This doctoral thesis has evolved from the publications given below. For each of the papers a designation to which chapter it belongs is provided. The main results were obtained during 2017 – 2022 and published in 31 peer-reviewed papers as well as 15 publications without review.

2016

1. I. Kabin, E. Alpirez Bock, C. Wittke, D. Kreiser, Z. Dyka, P. Langendörfer, "On the Influence of Hardware Technologies on the Vulnerability of Protected ECC Implementations", Proc. Session on Work in Progress, 19th EUROMICRO Conference on Digital Systems Design (DSD 2016), (2016), **(Sections 2.4.3, 4)**
2. Z. Dyka, E. Alpirez Bock, I. Kabin, P. Langendörfer, "Inherent Resistance of Efficient ECC Designs Against SCA Attacks", Proc. IFIP International Conference on New Technologies, Mobility and Security (NTMS 2016), (2016), **(Sections 2.4.1.1, 2.4.2, 2.4.3)**
3. Z. Dyka, E. Alpirez Bock, I. Kabin, P. Langendörfer, "Revisiting Random Permutation of Calculation Steps of a Field Multiplication as a DPA Countermeasure", Proc. 25th Crypto-Day, (2016), **(Section 6.1.2)**
4. I. Kabin, Z. Dyka, P. Langendörfer, "Influence of Multiplier on the Security of Elliptic Curve Cryptography Design", Proc. 25rd Crypto-Day, (2016), **(Section 6.1)**

2017

5. Z. Dyka, I. Kabin, D. Kreiser, P. Langendörfer, "Horizontal DPA Attacks: Low-Cost and High-Effective", Proc. 26th Crypto-Day, (2017), **(Section 3.2.3)**
6. I. Kabin, E. Vogel, Z. Dyka, P. Langendörfer, "DEMO: EMA as a Mean to Find Spatial and Time SCA Leakage Sources", Proc. International Conference on Reconfigurable Computing and FPGAs (ReCon-Fig 2017), (2017), **(Section 3.2.6)**
7. I. Kabin, Z. Dyka, D. Kreiser, P. Langendörfer, "Methods for Increasing the Resistance of Cryptographic Designs against Horizontal DPA Attacks", Proc. 19th International Conference on Information and Communications Security (ICICS 2017), (2017), **(Sections 6.1.1, 6.1.2)**
8. I. Kabin, Z. Dyka, D. Kreiser, P. Langendörfer, "Horizontal Address-Bit DPA against Montgomery *kP* Implementation", Proc. International Conference on Reconfigurable Computing and FPGAs (ReConFig 2017), (2017), **(Sections 2.2.1.1, 7.2.1)**
9. I. Kabin, Z. Dyka, D. Kreiser, P. Langendörfer, "Attack against Montgomery *kP* Implementation: Horizontal Address-Bit DPA? ", Proc. EUROMICRO Conference on Digital System Design (DSD/SEAA 2017), (2017), **(Sections 2.2.1.1, 7.2.1)**
10. I. Kabin, Z. Dyka, D. Kreiser, P. Langendörfer, "Evaluation of Resistance of ECC Designs protected by Different Randomization Countermeasures against Horizontal DPA Attacks", Proc. 15th IEEE East-West Design & Test Symposium (EWDTS 2017), 398 (2017), **(Section 5.1)**

11. I. Kabin, Z. Dyka, C. Wittke, D. Kreiser, P. Langendörfer, "Horizontal DEMA Attack as Low Cost Effective Mean to Reveal Keys", Proc. 26th Crypto-Day, (2017), **(Section 2.2.1.1)**
12. C. Wittke, I. Kabin, D. Kreiser, Z. Dyka, P. Langendörfer, "Selecting the Best Suitable EM Probe Using Horizontal DEMA Attack", Proc. 26th Crypto-Day, (2017), **(Section 3.1.3.3)**

2018

13. Z. Dyka, D. Kreiser, I. Kabin, P. Langendörfer, "Flexible FPGA ECDSA Design with a Field Multiplier Inherently Resistant against HCCA", Proc. International Conference on Reconfigurable Computing and FPGAs (ReConFig 2018), (2018), **(Sections 6.1.1.1, 8.1)**
14. I. Kabin, D. Kreiser, Z. Dyka, P. Langendörfer , "FPGA Implementation of ECC: Low-Cost Countermeasure against Horizontal Bus and Address-Bit SCA", Proc. International Conference on Reconfigurable Computing and FPGAs (ReConFig 2018), (2018), **(Section 6.2)**
15. I. Kabin, Z. Dyka, D. Kreiser, P. Langendörfer, "Improving DEMA Attack Results using Different Compression Methods", Proc. 27th Crypto-Day 2018, (2018), **(Sections 3.1.1, 3.2.1, 3.2.4, 3.2.6)**
16. I. Kabin, Z. Dyka, D. Kreiser, P. Langendörfer, "Unified Field Multiplier for ECC: Inherent Resistance against Horizontal SCA Attacks", Proc. 13th IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS 2018), (2018), **(Sections 2.2.1.1, 8.1)**
17. I. Kabin, Z. Dyka, D. Kreiser, P. Langendörfer, "Horizontal Address-Bit DEMA against ECDSA", Proc. 9th IFIP International Conference on New Technologies, Mobility & Security (NTMS 2018), (2018), **(Sections 2.2.1.1, 3.1.3, 7.2.1)**
18. I. Kabin, D. Kreiser, Z. Dyka, P. Langendörfer, "A Secure Scalable Dual-Field Multiplier for ECC", Proc. EUROMICRO Conference on Digital System Design and Software Engineering and Advanced Applications (Euromicro DSD & SEAA 2018) - Session on Work in Progress, (2018), **(Section 8.1)**
19. I. Kabin, Z. Dyka, D. Kreiser, P. Langendörfer, "Low-Cost Countermeasure Against Horizontal Bus and Address-Bit SCA", Proc. 29th Crypto-Day 2018, (2018), **(Section 6.2)**
20. I. Kabin, Z. Dyka, D. Kreiser, P. Langendörfer, "Low-Cost and High Efficient Horizontal Attacks against ECDSA", Proc. 27th Crypto-Day 2018, (2018), **(Section 2.2.1.1)**
21. D. Kreiser, Z. Dyka, I. Kabin, C. Wittke, P. Langendörfer, "HCCA against Montgomery *kP* Design", Proc. 29th Crypto-Day 2018, (2018), **(Section 6.1.1.1)**
22. E. Vogel, I. Kabin, Z. Dyka, P. Langendörfer, "Localized EMA as a Mean to Find Spatial and Time SCA Leakage Sources", Proc. 27th Crypto-Day 2017, (2018), **(Section 3.2.6)**

2019

23. Z. Dyka, I. Kabin, D. Klann, F. Vater, P. Langendörfer, "Caution: GALS-ification as a Means against SCA Attacks", Proc. 17th IEEE East-West Design & Test Symposium (EWDTS 2019), 97 (2019), **(Section 5.2)**
24. I. Kabin, Z. Dyka, D. Klann, P. Langendörfer, "Horizontal DPA Attacks against ECC: Impact of Implemented Field Multiplication Formula", Proc. 14th IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS 2019), (2019), **(Sections 6.1)**
25. I. Kabin, D. Klann, Z. Dyka, A. Datsuk, P. Langendörfer, "Demonstrating Horizontal Attacks using IHP's Side Channel Analysis Tool", Proc. International Conference on Reconfigurable Computing and FPGAs (ReConFig 2018), (2018), **(Section 3.2.6)**

26. I. Kabin, M. Aftowicz, Y. Varabei, D. Klann, Z. Dyka, P. Langendörfer, “Horizontal Attacks Using K-Means: Comparison with Traditional Analysis Methods”, Proc. 10th IFIP International Conference on New Technologies, Mobility & Security (NTMS 2019), (2019), **(Section 3.2.4)**
27. I. Kabin, Z. Dyka, D. Klann, P. Langendörfer, “Horizontal Attacks against ECC: From Simulations to ASIC”, Proc. International Workshop on Information & Operational Technology (IT & OT) Security Systems (IOSec 2019), (2019), **(Sections 6.4, 7.1.2)**
28. I. Kabin, Z. Dyka, D. Klann, P. Langendörfer, “On the Complexity of Attacking Commercial Authentication Products”, Proc. 10th IFIP International Conference on New Technologies, Mobility & Security (NTMS 2019), (2019), **(Section 2.1)**
29. I. Kabin, Z. Dyka, D. Klann, P. Langendörfer, “Fast and Secure Unified Field Multiplier for ECC Based on the 4-Segment Karatsuba Multiplication”, Proc. 17th IEEE East-West Design & Test Symposium (EWDTS 2019), 320 (2019), **(Section 8.1)**
30. I. Kabin, Z. Dyka, D. Klann, P. Langendörfer, “Influence of Synopsys Design Compiler Options on the Success of Horizontal DPA Attacks”, Proc. 22nd EUROMICRO Conference on Digital System Design and Software Engineering and Advanced Applications - Session on Work in Progress (Euromicro DSD & SEAA 2019), (2019), **(Section 6.4)**
31. I. Kabin, M. Aftowicz, D. Klann, Y. Varabei, Z. Dyka, D. Klann, P. Langendörfer, “Horizontal SCA Attack using Machine Learning Algorithms”, Proc. 30th Crypto Day Matters 2019, (2019), **(Section 3.2.4)**
32. A. Sosa, I. Kabin, Z. Dyka, D. Klann, P. Langendörfer, “On the Impact of the Sampling Rate on the Success of Horizontal DEMA Attack”, Proc. 31st Crypto Day Matters 2019, (2019), **(Section 3)**
33. Y. Varabei, I. Kabin, Z. Dyka, D. Klann, P. Langendörfer, “Intelligent Clustering as a Means to Improve K-Means Based Horizontal Attacks”, Proc. 30th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2019) Workshop-W6: Machine Learning for Security and Cryptography, (2019), **(Section 3.2.4)**
34. C. Wittke, I. Kabin, D. Klann, Z. Dyka, A. Datsuk, P. Langendörfer, “Horizontal DEMA Attack as the Criterion to Select the Best Suitable EM Probe”,e-print archive https://eprint.iacr.org/2018/1181.pdf, **(Section 3.1.3.3)**

2020

35. Dan Klann, Marcin Aftowicz, Ievgen Kabin, Zoya Dyka and Peter Langendoerfer, “Integration and Implementation of four different Elliptic Curves in a single high-speed Design considering SCA”, Proc. of DTIS-2020, **(Section 8.1)**
36. Ievgen Kabin, Zoya Dyka, Marcin Aftowicz, Dan Klann and Peter Langendoerfer, “Resistance of the Montgomery *kP* Algorithm against Simple SCA: Theory and Practice”, Proc. of LATS-2020, **(Section 3.2.5.1)**
37. Ievgen Kabin, Zoya Dyka, Dan Klann and Peter Langendoerfer, “Methods Increasing Inherent Resistance of ECC Designs against Horizontal Attacks”, Integration, 2020, **(Sections 4, 6.3)**
38. Marcin Aftowicz, Ievgen Kabin, Dan Klann, Yauhen Varabei, Zoya Dyka and Peter Langendoerfer, “Horizontal SCA Attacks against *kP* Algorithm Using K-Means and PCA”, Proc. of MECO-2020, **(Section 3.2.4)**
39. I. Kabin, Z. Dyka, D. Klann, N. Mentens, L. Batina and P. Langendoerfer, "Breaking a fully Balanced ASIC Coprocessor Implementing Complete Addition Formulas on Weierstrass Elliptic Curves," 2020 23rd Euromicro Conference on Digital System Design (DSD), 2020, pp. 270-276, doi: 10.1109/DSD51259.2020.00051, **(Section 5.3)**

40. I. Kabin, Z. Dyka and P. Langendoerfer, "Automated Simple Analysis Attack," 2020 9th Mediterranean Conference on Embedded Computing (MECO), 2020, pp. 1-4, doi: 10.1109/MECO49872.2020.9134160, **(Sections 3.2.5.1, 3.2.6)**

2021

41. I. Kabin, Z. Dyka, D. Klann, P. Langendörfer, „EC Scalar Multiplication: Successful Horizontal Address-Bit SCA against Atomic Patterns", Latin-American Test Symposium (LATS 2021), **(Section 8.2)**
42. M. Aftowicz, I. Kabin, Z. Dyka, P. Langendörfer, „Clustering versus Statistical Analysis for SCA: when Machine Learning is Better", 10th Mediterranean Conference on Embedded Computing (MECO), 2021, pp. 1-5, doi: 10.1109/MECO52532.2021.9460161, **(Section 3.2.4)**
43. Z. Dyka, I. Kabin, D. Klann, P. Langendörfer, „Multiplier as a Mean for Reducing Vulnerability of Atomic Patterns to Horizontal Address-Bit Attacks",10th Mediterranean Conference on Embedded Computing (MECO), 2021, pp. 1-6, doi: 10.1109/MECO52532.2021.9460158, **(Section 8.4)**
44. I. Kabin, D. Klann, Z. Dyka, P. Langendörfer, „Fast Dual-Field ECDSA Accelerator with Increased Resistance against Horizontal SCA Attacks",IEEE International Conference on Cyber Security and Resilience (CSR), 2021, pp. 273-280, doi: 10.1109/CSR51186.2021.9527912, **(Section 8.1)**
45. I. Kabin, Z. Dyka, D. Klann, M. Aftowicz and P. Langendoerfer, "FFT based Horizontal SCA Attack against ECC," 2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2021, pp. 1-5, doi: 10.1109/NTMS49979.2021.9432665, **(Sections 3.2.5.2, 7.1.2, 7.2.1)**

2022

46. I. Kabin, Z. Dyka, and P. Langendoerfer, "Atomicity and Regularity Principles Do Not Ensure Full Resistance of ECC Designs against Single-Trace Attacks," Sensors, vol. 22, no. 8, Art. no. 8, Jan. 2022, doi: 10.3390/s22083083, **(Section 8)**

2 Background

In this section we do not give a full overview of EC point operations, field operations that are the mathematical basics for ECC, etc. It was done in many scientific papers, journals and books, for example in [1], [3]-[7] and is not the topic of this research. We explain here only the details that are necessary for understanding the ideas proposed and evaluated in this thesis as well as the results presented in this work.

2.1 *kP*: mostly attacked operation

The main operation in ECC is the multiplication of an EC point $\boldsymbol{P}$ with a scalar k, denoted as $k\boldsymbol{P}$ operation and often known as an EC point multiplication or as an EC scalar multiplication. $k\boldsymbol{P}$ operation is defined as a k-times addition of the EC point $\boldsymbol{P}$ to itself:

$$k\boldsymbol{P} = \underbrace{\boldsymbol{P}+\boldsymbol{P}+\boldsymbol{P}+\ldots+\boldsymbol{P}}_{k\text{-times}} \tag{1}$$

The EC point $\boldsymbol{P}$ is a point on an EC over a finite field. The coordinates of each EC point are usually represented as two long binary numbers, up to the maximum key length: $\boldsymbol{P}=(x, y)$. EC point doubling $2\boldsymbol{P}$ and point addition $\boldsymbol{P}+\boldsymbol{Q}$ are defined as a sequence of mathematical operations in finite fields, called also Galois fields (GF) in honour of Evariste Galois (a French mathematician, 25.10.1811 – 31.05.1832).

The security of cryptographic approaches is based on the secrecy of the private key. The scalar k is the private key if EC-based authentication is performed, see section 2.1.1. In the case of EC digital signature generation corresponding to the Elliptic Curve Digital Signature Algorithm (ECDSA) protocol the private key can be easily calculated, if the scalar k used in a $k\boldsymbol{G}$ point multiplication is known, see section 2.1.2. The scalar k is usually about 200 bits. Until 2030 the scalar k has to be up to 256 bits long. Starting from 2031 the NIST EC *P-256* [8] and up to 283 bits long if the NIST EC *B-283* [8] has to be used [9]. Their "security level" is comparable with 3072-bit long RSA private key or with 256-bit long keys for standardized symmetric cryptographic approaches, for example AES.

The goal of many SCA attacks is to reveal the private key using any information available about the process of the EC scalar multiplication. For example, the time of executions, the energy consumption or the electromagnetic emanation measured while performing the EC scalar multiplication can be analysed to reveal the scalar used in the operation.

Resistance against SCA attacks is a very important feature for cryptographic chips. If the scalar k used in the $k\boldsymbol{P}$ operation can be extracted by an attacker, the attacker can falsify its identity. Revealing the scalar k processed an ECDSA signature generation causes the same consequences, i.e. the loss of the identity. Due to the importance of the secrecy of the processed scalar k, we denote it in the rest of this work as the *key k*.

2.1.1 Authentication

The Diffie-Hellman protocol (DH) was proposed in 1976 and exploits modular exponentiation in finite fields to share a secret between two participants. The modular exponentiation is a time-consuming and processing intense operation. In [3] Neal Koblitz proposed to use the $k\boldsymbol{P}$ operation instead of the modular exponentiation. This protocol is called Elliptic Curve Diffie-Hellman (ECDH). Alice and Bob use their public knowledge about an elliptic curve

over a finite field to generate a common secret. Each of the participants generates a random number once and performs the $k\boldsymbol{P}$ operation twice (see TABLE I.).

TABLE I. GENERATION OF A SHARED SECRET USING ECDH

A (Alice)	***B*** (Bob)
knows point $\boldsymbol{G}$ of an EC	knows point $\boldsymbol{G}$ of an EC
• generates a random a	• generates a random b
• sends to ***B***: $\boldsymbol{Q_A} = a \cdot \boldsymbol{G}$	• sends to ***A***: $\boldsymbol{Q_B} = b \cdot \boldsymbol{G}$
• receives $\boldsymbol{Q_B}$	• receives $\boldsymbol{Q_A}$
• calculates $\boldsymbol{Q} = a \cdot \boldsymbol{Q_B}$	• calculates $\boldsymbol{Q} = b \cdot \boldsymbol{Q_A}$
• $\boldsymbol{secret_{AB}}$=***x-coord of Q***	• $\boldsymbol{secret_{AB}}$=***x-coord of Q***

The ECDH protocol works without using public cryptographic keys of the participants. If the knowledge of the public key of a participant is used, the ECDH approach allows mutual authentication of both participants or of only one of the participants. For example, Alice can authenticate Bob using the knowledge of Bob's public key. To do so Alice sends a request for his certificate to Bob. Bob answers to Alice, i.e. he sends to Alice his certificate that contains the public key of Bob signed by a trusted authority. Alice verifies the certificate. Thereafter Alice is sure that Bob is the owner of the public key $\boldsymbol{Pub_B}$. To be sure that her communication partner is really Bob Alice generates a random number r, calculates two elliptic curve point multiplications: $\boldsymbol{Q}=r \cdot \boldsymbol{Pub_B}$ and $\boldsymbol{R}=r \cdot \boldsymbol{G}$, where $\boldsymbol{G}$ is the base point of the EC and sends point $\boldsymbol{R}$ to Bob (see TABLE II.).

TABLE II. AUTHENTICATION BASED ON ECDH APPROACH

A (Alice)	***B*** (Bob)
knows point $\boldsymbol{G}$ of an EC knows public key of Bob $\boldsymbol{Pub_B}$	knows point $\boldsymbol{G}$ of an EC **B** is owner of $\boldsymbol{(k_B; Pub_B)}$
• generates a random r	
• calculates $\boldsymbol{Q} = r \cdot \boldsymbol{Pub_B}$	
• sends to ***B***: $\boldsymbol{R} = r \cdot \boldsymbol{G}$	• receives $\boldsymbol{R}$
• receives $\boldsymbol{Q_B}$	• sends to ***A***: $\boldsymbol{Q_B} = k_B \cdot \boldsymbol{R}$
• if $\boldsymbol{Q} = \boldsymbol{Q_B}$ authentication is **ok**	

Bob calculates the elliptic curve point multiplication $k_B \cdot \boldsymbol{R}$, where $\boldsymbol{R}$ is the EC point received and k_B is his private key and sends $\boldsymbol{Q_B}$ the result of the calculation to Alice. Alice compares the received EC point $\boldsymbol{Q_B}$ with the calculated point $\boldsymbol{Q}$. The authentication passes if both points are equal. Please note that $\boldsymbol{Q_B} = k_B \cdot \boldsymbol{R} = k_B \cdot r \cdot \boldsymbol{G}$ and $\boldsymbol{Q} = r \cdot \boldsymbol{Pub_B} = r \cdot k_B \cdot \boldsymbol{G}$, i.e. the EC point $\boldsymbol{Q}$ is equal to the EC point $\boldsymbol{Q_B}$ only if Bob is the owner of the key pair $\boldsymbol{(k_B; Pub_B)}$.

All operations on Bob's side can be implemented in hardware as an authentication chip. The ECs over binary extended fields $GF(2^n)$ such as the standardized NIST curves [8] *B-163*, *B-233*, *B-283* are especially suitable for hardware implementations. This is due to the fact that the field operations such as addition and multiplication are done without carry-bit propagation. This reduces the area of authentication chips significantly leading to low manufacturing costs for authentication chips. Authentication chips from different manufacturers are available on the market, for example [10]-[12]. The typical application is the authentication of devices or their parts, for example confirmation of the originality of printer cartridges, electronic accessories such as AC/DC adapters, cables, keyboards, docking stations, batteries, digital headsets, electronic cigarettes etc.

The use of the Montgomery $k\boldsymbol{P}$ algorithm in Lopez-Dahab projective coordinates [5] allows to perform the calculation with just the x-coordinate of the input EC point. This helps to reduce the execution time, area and energy consumption of authentication significantly. Recovering and sending of the y-coordinate of the results can be saved additionally. Due to these facts, the Montgomery $k\boldsymbol{P}$ algorithm in Lopez-Dahab projective coordinates is the mostly used algorithm for implementing EC scalar multiplications using ECs over binary extended fields $GF(2^n)$. All these "tricks" are exploited in industrial authentication chips, for example in the

NXP A1006 Secure Authenticator [10], [13] and probably also in the Optiga Trust B secure solution by Infineon [11].

Resistance of the Montgomery $k\boldsymbol{P}$ algorithm in Lopez-Dahab projective coordinates against SCA attacks is a very important feature required for authentication chips.

2.1.2 ECDSA

The Elliptic Curve Digital Signature Algorithm (ECDSA) is an algorithm to generate digital signatures as described in [8] and [14]. A signature generation requires a single EC scalar multiplication and a verification of signature requires two $k\boldsymbol{P}$ operations. The $k\boldsymbol{P}$ operations can be implemented as a sequence of mathematical operations in finite fields. Additionally, a hash function has to be calculated. The signature generation and verification algorithms are sketched in TABLE III.

Corresponding to the ECDSA algorithm, public keys are points of an EC and private keys are big binary numbers, smaller than the order ε of the base point $\boldsymbol{G}$. ε and $\boldsymbol{G}$ are parameters of the EC that was selected for cryptographic operations; they are given in [8]. To generate the signature Alice calculates the hash **e** of her message and performs the multiplication of the base point $\boldsymbol{G}$ with a generated random scalar k (see step 4 in TABLE III.), i.e. an EC point multiplication, usually denoted as $k\boldsymbol{P}$ operation. The result of the performed $k\cdot\boldsymbol{G}$ operation is also a point of the EC. Its x-coordinate is part of the digital signature, denoted as r. Using the calculated r and **e**, the generated scalar k and the private key key_A Alice calculates the last part of the signature, denoted as s (see step 6 in TABLE III.). Alice sends her message to Bob with its digital signature, i.e. with two numbers: r and s.

TABLE III. GENERATION AND VERIFICATION OF SIGNATURE CORRESPONDING TO ECDSA

Alice has the ECC key pair ($\boldsymbol{Pub_A}$; key_A)	Bob knows Alice's public key $\boldsymbol{Pub_A}$
Signature generation	Signature verification
1. writes a message 2. **e** = hash(message) 3. generates random $k < \varepsilon$ 4. $T = (x_T, y_T) = k\cdot\boldsymbol{G}$ 5. $r = x_T \mod \varepsilon$ 6. $s = (\mathbf{e}+r\cdot key_A)/k \mod \varepsilon$ 7. sends to Bob: (message, r, s)	1. receives: (message, r, s) 2. **e** = hash(message) 3. $u_1 = \mathbf{e}/s \mod \varepsilon$ 4. $u_2 = r/s \mod \varepsilon$ 5. $T = (x_T, y_T) = u_1\cdot\boldsymbol{G} + u_2\cdot\boldsymbol{Pub_A}$ 6. check if: $r = x_T \mod \varepsilon$

Bob receives the signed message from Alice and verifies the signature. To do this Bob needs to know Alice's public key $\boldsymbol{Pub_A}$. Bob calculates the hash **e** of the received message, scalar u_1 and scalar u_2 (see steps 2–4 in TABLE III.). Then Bob performs two EC point multiplications and one point addition to calculate the point $\boldsymbol{T}$ as shown in step 5 in TABLE III. If the x-coordinate of this point is equal to the received r, the signature was successfully verified. For the verification Bob doesn't use his own key pair $\boldsymbol{Pub_B}$ and key_B.

For a signature generation the EC scalar multiplication $k\boldsymbol{G}$ is the critical operation. It is the multiplication of the EC basis point $\boldsymbol{G}$ with a random number k. But if an attacker can reveal this random scalar k, the private key Key used for a signature generation can be easy calculated as follows[1]:

$$Key = \frac{s\cdot k - \mathbf{e}}{r} \bmod \varepsilon \tag{2}$$

[1] This fact is known as the "security requirements for per-message secrets", see [14], p.34.

Numbers *r, s* and the message itself are transmitted to a receiver, i.e. the attacker knows these numbers and the message. Thus, the attacker can also calculate the hash value **e** of the message. The point $\boldsymbol{G}$ and its order ε are public parameters of the EC, i.e. they are known to the attacker, as well.

As a consequence, the identity of the signer can be stolen. This means that implementations of the $k\boldsymbol{P}$ operation, at least for the signature generation process, have to be resistant against SCA attacks.

2.2 SCA Attacks

2.2.1 Attacks relevant for this work

In this section we give an overview of attacks against binary $k\boldsymbol{P}$ algorithms, i.e. algorithms processing the scalar k bit-by-bit. We concentrate on horizontal attacks applying statistical methods for the analysis, as well as on the attacks exploiting key-dependent addressing of the registers for extraction of the scalar k. We discuss here also known countermeasures against address-bit differential power analysis (DPA) attacks.

2.2.1.1 Horizontal attacks applying statistical analysis

Horizontal attacks are serious threats for ECC protocols. Practical examples of horizontal attacks against asymmetric cryptographic approaches such as RSA or ECC are the simple power analysis (SPA) attacks described in [15], simple electromagnetic analysis (SEMA) attacks e.g. [16], the Big Mac attack [17], the localized electromagnetic analysis (EMA) attack [18], horizontal collision correlation analysis (HCCA) attacks [19]-[20], horizontal DPA and differential electromagnetic analysis attacks [21]-[27], automated SPA attack [28] as well as single-trace attacks using statistical analysis in the frequency domain [29] or clustering methods for the analysis in the time domain [30]-[35].

The history of advanced single-trace attacks, i.e. the attacks using at least statistical methods for the analysis of traces, began in 1999 with the work published by T. Messerges, E. Dabbish and R. Sloan [36]. In this paper the authors concentrated on different vertical power analysis attacks against a smartcard implementation of modular exponentiation for RSA. But the authors also described their experiments for the determining and distinguishing the operation profiles of multiplication and squaring. In a measured power trace of a modular exponentiation a part corresponding to a multiplication was selected. Using the selected profile a cross-correlation was performed for the whole measured power trace. This method allowed partitioning of the trace into slots, where either a multiplication or a squaring operation was performed, but it was not possible to determine, which of the slots referred to multiplications, i.e. multiplications were not distinguished from the squaring operations. The equipment used for the measurements in [36] is not described. Due to the fact, that the equipment for the measurements significantly progressed in the last 20 years, we assume that the described horizontal cross-correlation attack can be successful using a modern high-end oscilloscope.

The next well-known horizontal attack applying statistical methods for the analysis was Walter's Big Mac attack [17]. It was performed in 2001 against an RSA implementation. The design consists of a single m x m bit multiplier for the calculation of n-bit long operands that are elements of a prime finite field, with $n > m$. The m x m bit multiplier requires only a single clock cycle for a partial product calculation. The attacked algorithm is a binary square-and-multiply algorithm. This algorithm requires a squaring operation for processing a key bit value 0 and two operations – a squaring and a multiplication – for processing a key bit value 1. The processed key is the private exponent and the goal of the attacker is to reveal it. If the squaring

operation is implemented as a multiplication with the same operands, both operations require the same number of clock cycles for the calculation and accumulation of all partial products. In this case the power shapes of multiplications and squaring operations look similar, i.e. the private exponent key cannot be revealed via visual inspection of the traces. However, it is still possible to reveal the private exponent based on the fact that the energy consumptions of multiplications with a common operand a_i are more similar to each other than those of the multiplications with different operands. Averaging of the energy consumption (or the power shapes) of the multiplications with a common operand, for example for the multiplications *ab*, *ac*, *ad*, reduces the influence of the different operands *b*, *c* and *d*. Their influence can be observed as a kind of a noise that is reduced due to the averaging. But the influence of the operand *a* is not reduced, i.e. the "contribution" of the operand *a* will be "more visible" after averaging. In case of multiplications with completely different operands the averaging reduces the contribution of all operands. Thus, if it is possible to select 2 groups of multiplications with a common operand, for example a group *ab*, *ac*, *ad*, and a group *ae*, *af*, *ag*, the averaged power shapes of each group will be more similar to each other than to the one of the group *ab, cd, ef*.

We explain how this assumption can be applied to distinguish the multiplication with two equal operands, i.e. the squaring operation, from the multiplication with two different operands. We use the example of 3-segment long operands multiplication, i.e. $A = a_2 a_1 a_0$ and $B = b_2 b_1 b_0$. Each segment a_i, b_i is *m*-bit long.

$$\begin{aligned} A^2 = A \cdot A &= \left(a_2 2^{2m} + a_1 2^m + a_0\right) \cdot \left(a_2 2^{2m} + a_1 2^m + a_0\right) = \\ &= \underset{pp1}{a_2 a_2\, 2^{4m}} + \underset{pp2}{a_2 a_1\, 2^{3m}} + \underset{pp3}{a_2 a_0\, 2^{2m}} + \underset{pp4}{a_1 a_2\, 2^{3m}} + \underset{pp5}{a_1 a_1\, 2^{2m}} + \underset{pp6}{a_1 a_0\, 2^m} + \underset{pp7}{a_0 a_2\, 2^{2m}} + \underset{pp8}{a_0 a_1\, 2^m} + \underset{pp9}{a_0 a_0} \end{aligned} \tag{3}$$

Due to the assumption described above, the average power profile of the partial products *pp1*, *pp2* and *pp3* with the common operand a_2 is similar to the average power profile of the partial products *pp1*, *pp4* and *pp7*, with the same common operand a_2.

Additionally, the following sets of the partial products can be investigated:

- the average power profile of the partial products *pp4*, *pp5* and *pp6* is similar to the average power profile of the partial products *pp2*, *pp5* and *pp8*;
- the average power profile of the partial *products pp3*, *pp6* and *pp9* is similar to the average power profile of the partial products *pp7*, *pp8* and *pp9*.

If we observe a multiplication of two different operands, we cannot find the sets with a similar averaged power profile, due to the fact that all partial products $a_i b_j$ are here different:

$$\begin{aligned} A \cdot B &= \left(a_2 2^{2m} + a_1 2^m + a_0\right) \cdot \left(b_2 2^{2m} + b_1 2^m + b_0\right) = \\ &= \underset{pp1}{a_2 b_2\, 2^{4m}} + \underset{pp2}{a_2 b_1\, 2^{3m}} + \underset{pp3}{a_2 b_0\, 2^{2m}} + \underset{pp4}{a_1 b_2\, 2^{3m}} + \underset{pp5}{a_1 b_1\, 2^{2m}} + \underset{pp6}{a_1 b_0\, 2^m} + \underset{pp7}{a_0 b_2\, 2^{2m}} + \underset{pp8}{a_0 b_1\, 2^m} + \underset{pp9}{a_0 b_0} \end{aligned} \tag{4}$$

Thus, if an attacker knows the sequence of the partial product calculations, he can calculate the average profile of the selected partial products and use it to determine if a multiplication or a squaring operation was performed. The revealed sequence of the multiplications and squaring operations can be used to easily reveal the private key. This attack exploits the key-dependent data processed by the field multiplier and, therefore, it represents a data-bit attack.

In 2010, Clavier et al. [37] presented a horizontal correlation analysis attack against an RSA decryption, i.e. against an RSA exponentiation of a message using a private key. The authors mentioned that the attack can be successfully applied against regular exponentiation algorithms such *square-and-multiply-always* as well as against the Montgomery ladder, even

when implementing the atomicity principle [38]. The fact, that the key blinding technique is not effective against horizontal attacks, was discussed as well. The attack requires additionally a single trace of an RSA encryption (using a public key) of a known message and is based on the assumption, that the first *s* bits of the private exponent processed in the attacked trace, are already known to the attacker. Authors of [37] introduced as well the classification of SCA attacks into vertical and horizontal.

Walter's ideas [17] about the distinguishability of multiplications calculating the same and completely different operands were applied in [20] against an EC implementation protected by the atomicity principle. The attack is known as Horizontal Collision Correlation Attack[2]. This attack exploits the following observations:

- *Observation1*: two multiplications with one common operand can be distinguished from two multiplications with completely different operands, i.e. power shapes of the multiplications *ab* and *ac* (with the same operand *a*) are more similar to each other than the power shapes of the multiplications *ab* and *cd*.
- *Observation2:* the EC point addition implemented using atoms proposed in [38], [39]-[40] has two multiplications with the same operand, such as *ab* and *ac*, but this is not the case for the multiplications in EC point doubling.

These observations can be used for revealing the key because a point doubling can be distinguished from a point addition even in *double-and-add* $k\boldsymbol{P}$ algorithms obeying the atomicity principle.

Please note that the field multiplier analysed in [20] was implemented using the classical multiplication method (MM) and experimental results presented in [20] confirm the basic observations. Pearson's coefficients calculated for traces of two multiplications with a common operand differ significantly from coefficients calculated for two multiplications with different operands.

We performed this type of attack against the Montgomery $k\boldsymbol{P}$ algorithm for binary ECs. The scalar *k* cannot be revealed in this case, since the *Observation2* is not applicable for the Montgomery ladder. But this type of attack can help to separate measured traces into slots, i.e. into parts corresponding to the processing of a single key bit of the scalar *k*. This is due to the fact that each slot contains a multiplication with the EC parameter *b* as well as a multiplication with the *x* coordinate of the input point $\boldsymbol{P}$. The parameter *b* is a publicly known constant value. The coordinate *x* (or its randomised value *X*) is constant during a single $k\boldsymbol{P}$ calculation.

The separation of traces into slots is a preparation step for horizontal attacks [27]. It can be done using this type of attack or using a cross-correlation analysis, as in [17]. We investigated the correlation between multiplications with the goal to evaluate the applicability of the *Observation1* to our field multiplier implementing the 4-segment Karatsuba MM. *Observation1* was not confirmed for the investigated field multiplier, due to the length of its partial operands and significantly smaller number of the calculated partial products, compared to the classical MM[3]. We were able to separate the attacked $k\boldsymbol{P}$ traces into slots calculating the Pearson correlation coefficients, of 9 clock cycles long shapes, corresponding to a single multiplication. Not the field multiplier but other operations performed in parallel to the multiplications are reasons for the successful separation. We described our experiments in [41]-[43]. We do not give more details here, because these investigations are relevant for data-bit attacks and do not influence the results of address-bit attacks described in this work.

[2] A similar attack against RSA is described in [19].

[3] The 4-segment Karatsuba MM requires a calculation of 9 partial products. The classical MM requires 16 partial products, for the same segmentation of the multiplicands. Thus, field multipliers implementing the 4-segment Karatsuba MM can reduce the execution time of a $k\boldsymbol{P}$ calculation significantly, i.e. about 40%.

2.2.1.2 Vertical address-bit attacks and countermeasures

The horizontal attacks described above belong to the group of data-bit attacks. In this section we overview 3 papers [44]-[46], reporting on successful address-bit DPAs against ECC and countermeasures introduced.

The addressing of registers was already earlier identified as a leakage that allows to successfully distinguish whether a key bit '1' or '0' was processed. E.g. Itoh et al. [44] presented a successful vertical, address-bit DPA attack against ECC, already in 2002. We describe this vertical address-bit DPA attack here in more detail due to the fact that the phenomenon causing the high success rate of our horizontal address-bit SCA attacks against IHP $k\boldsymbol{P}$ designs is very similar to the one described in [44].

The vertical address-bit DPA attack [44] was performed against a hardware implementation of the following variant of the Montgomery ladder:

Algorithm 1:Montgomery *kP* algorithm corresponding to [44]

1. $\boldsymbol{Q_0 \leftarrow P;\ Q_1 \leftarrow 2P;}$
2. **for** $i = (l-2)$ **downto** 0
3. $\quad \boldsymbol{Q_2 \leftarrow 2\ Q_{k_i}}$
4. $\quad \boldsymbol{Q_1 \leftarrow Q_0 + Q_1}$
5. $\quad \boldsymbol{Q_0 \leftarrow Q_{2-k_i}}$
6. $\quad \boldsymbol{Q_1 \leftarrow Q_{1+k_i}}$
7. Output $\boldsymbol{Q_0 = k \cdot P}$

Two attacks were performed:

Attack 1:

500 traces for a $k\boldsymbol{P}$ execution with a given scalar k=1111…1 were measured and averaged. Additionally, 500 traces of a $k\boldsymbol{P}$ execution with an unknown scalar d were measured. Thus, 1000 traces were measured in this attack to reveal the key. The EC points for all these measurements were randomly chosen but the scalars in each set of 500 traces were constant.

The dependency of the data on the processed scalar in the measured power traces was significantly reduced using different EC points $\boldsymbol{P}$ and averaging of the traces. The key-dependent addressing of registers in the Montgomery $k\boldsymbol{P}$ algorithm in step 3, 5 and/or 6 was not influenced by averaging the data. Thus, the difference of the averaged traces has spikes in the slots corresponding to the processing of an i^{th} bit of the scalar if $k_i \neq d_i$. Such spikes are good observable and allow to reveal the unknown scalar d.

Attack 2:

500 traces of a dP execution with an unknown but constant scalar d were measured and averaged. The traces were partitioned into slots. Each slot S_i corresponds to the processing of a single scalar's bit d_i.

Differences of the power shapes $D_{i,0}=S_i - S_0$ were calculated for each $i>0$. Spikes in a shape $D_{i,0}$ mark the fact that the bit value d_i differs from the bit value d_0, i.e. it marks $d_i \neq d_0$. This kind of the comparison can be performed not only with the slot S_0 but also using all other slots pairwise. Using these good observable spikes in the difference of the power shapes the processed scalar was successfully revealed, too.

Authors of [44] discovered that an implementation remains vulnerable to their attack even after applying Coron's countermeasures such as randomization of projective coordinates of the input, i.e. EC point P. In [45] only the scalar splitting ($d=(d-r)+r$, with random r) proposed by [47] was mentioned as possible countermeasure for binary $k\boldsymbol{P}$ algorithms. Additionally, in [45] the randomization of address bits while accessing registers was proposed as a countermeasure against vertical address-bit DPA attack due to the significant time overhead of the splitting. Later in 2010, the authors of [46] showed the ineffectiveness of the countermeasures presented in [45] and proposed an improved randomisation of the addresses of the registers.

Please note that the key randomization mentioned in [44] as an effective countermeasure against the performed vertical attack is no longer effective against an address-bit DPA if the attack can be performed horizontally. The randomizations of addressing presented in [45] and [46] slightly increase the effort for revealing a key compared to unprotected Montgomery ladder implementations but they do not prevent or reduce the success of single-trace attacks.

2.3 Countermeasures: basic principles

In many algorithms, the scalar k is processed bitwise, especially in hardware implementations of $k\boldsymbol{P}$ accelerators. Processing a key bit value '0' requires only an EC point doubling operation, but processing a key bit value '1' requires an EC point doubling and an EC point addition operation corresponding to the binary *left-to-right* and *right-to-left double-and-add* $k\boldsymbol{P}$ algorithms that are the oldest, well-known and simplest EC point multiplication algorithms. Both algorithms are given below for comparison. They differ in the sequence of the processing of the key bits, in the initialization step (see step 1 of both algorithms), and in the sequence of the EC point operations. In the *right-to-left* algorithm, an EC point addition is performed if the currently processed key bit value is '1'. An EC point doubling is calculated for each key bit, independent from its value. In the *left-to-right* algorithm, at first an EC point doubling is performed for each key bit followed by an EC point addition if the currently processed key bit value is '1'.

Algorithm 2: binary *double-and-add right-to-left* algorithm from k_0 to k_{l-1}

1. $\boldsymbol{Q_0 \leftarrow O;\ Q_1 \leftarrow P}$
2. **for** $i=0$ **to** $(l-1)$
3. $\quad$ ***if*** $k_i = 1$ then $\boldsymbol{Q_0 \leftarrow Q_0 + Q_1}$
4. $\quad$ $\boldsymbol{Q_1 \leftarrow 2Q_1}$
5. Output $\boldsymbol{Q_0 = k \cdot P}$

Algorithm 3: binary *double-and-add left-to-right* algorithm: from k_{l-1} to k_0

1. $\boldsymbol{Q_0 \leftarrow P;}$
2. **for** $i=(l-2)$ **downto** 0
3. $\quad$ $\boldsymbol{Q_0 \leftarrow 2Q_0}$
4. $\quad$ ***if*** $k_i = 1$ then $\boldsymbol{Q_0 \leftarrow Q_0 + P}$
5. Output $\boldsymbol{Q_0 = k \cdot P}$

An EC point doubling and an EC point addition are different operations and usually consist of different sequences of field operations, i.e. field additions, squaring operations, field multiplications and field divisions. Thus, the calculation of a point doubling differs significantly from a point addition. The power shapes of these operations also differ significantly. Consequently, the power profiles of a point doubling can be distinguished from power profiles of a point addition. Due to the fact that processing of a key bit value '0' requires only a point doubling and processing a key bit value '1' requires a point doubling and a point addition, the differences in the power trace can be easily used for revealing the key. This is the main disadvantage of both algorithms, i.e. they are vulnerable to simple side-channel analysis attacks. If the sequence of EC point operations can be determined, the scalar k (i.e. the key) can be revealed.

The designers can reduce the success of SCA attacks by making the shape of power profiles for processing all key bits independent of the processed key bit values.

The main idea of the countermeasures is to ensure that the processing of different key bits is indistinguishable. There are different ways to implement it, for example applying either regularity/atomicity principles or different randomization techniques.

2.3.1 Regularity and atomicity

When applying the regularity principle designers implement the same sequence of operations for the processing of each key bit. A simple way to make an algorithm regular is executing dummy operations, for example, a point doubling and a point addition can be calculated always for processing of each key bit value. If the processed key bit value is '0', the result of the point addition may not be used, i.e. it is a dummy operation that is necessary only for making the shape of the measured trace for the processing of a key bit '0' indistinguishable from the one of processing of a key bit '1'. The binary *double-and-add-always* algorithm proposed by Coron in 1999 [48] processes the scalar k bitwise from *left-to-right* as follows:

Algorithm 4: binary *double-and-add-always* (*left-to-right*)

1. $\boldsymbol{Q_0 \leftarrow P;}$
2. **for** $i = (l-2)$ **downto** 0
3. $\quad \boldsymbol{Q_0 \leftarrow 2Q_0}$
4. $\quad \boldsymbol{Q_1 \leftarrow Q_0 + P}$
5. $\quad \boldsymbol{Q_0 \leftarrow Q_{k_i}}$
6. Output $\boldsymbol{Q_0 = k \cdot P}$

The disadvantage of dummy operations is a (significantly) increased execution time and energy consumption.

For many years and even still nowadays this algorithm is considered as an effective means against SPA because the sequence of instructions in the algorithm does not depend on the processed key bit value, i.e. the algorithm is regular.

The weakness of this algorithm is step 5: $\boldsymbol{Q_0 \leftarrow Q_{k_i}}$. The power consumption of a write-to-register operation is proportional to the Hamming distance of the value stored in the register and the value to be stored. So, if the processed key bit value is $k_i = 0$, this operation is $\boldsymbol{Q_0 \leftarrow Q_0}$, i.e. it doesn't consume any energy. If the processed key bit value is '1', this operation is $\boldsymbol{Q_0 \leftarrow Q_1}$, i.e. it consumes energy proportional to Hamming distance of the values stored in the registers $\boldsymbol{Q_0}$ and $\boldsymbol{Q_1}$. This energy is small but can be detected for sure using statistical analysis. Depending on the implementation details of other blocks, for example if the multiplier is small or operations are not implemented in parallel, these differences can be exploited for successfully revealing the key by simple visualisation, i.e. designs based on the *double-and-add-always* algorithm are not implicitly protected against SCA attacks.

To improve the resistance of the *double-and-add-always* algorithm against SCA attacks a dummy write to register operation was proposed in [49] as follows:

Algorithm 5: binary *double-and-add-always-improved-symmetry (left-to-right)*

1. $\boldsymbol{Q_0 \leftarrow P;}$
2. **for** $i = (l-2)$ **downto** 0
3. $\quad \boldsymbol{Q_0 \leftarrow 2Q_0}$
4. $\quad \boldsymbol{Q_1 \leftarrow Q_0 + P}$
5. $\quad$ ***if*** $k_i = 1$ then $\boldsymbol{Q_0 \leftarrow Q_1}$
6. $\quad$ ***else*** $\boldsymbol{Q_1 \leftarrow Q_0}$
7. Output $\boldsymbol{Q_0 = k \cdot P}$

The Hamming distance of the registers $\boldsymbol{Q_0}$ and $\boldsymbol{Q_1}$ defines the power consumption of the data storing operation into register $\boldsymbol{Q_0}$ in step 5 as well as the dummy data storing operation in step 6. Thus, the data stored into the registers can no longer be used to determine which key bit value – '1' or '0' – is currently processed.

Regularity is the next idea proposed to make point doublings indistinguishable from point additions. If the power profile of a point doubling is the same as the one of a point addition, an attacker – theoretically – does not know, where a single point doubling for processing a key bit '0' was performed and where a point doubling and point addition for processing a key bit '1'. Different unified formulae for calculating point doublings and point additions using the same sequence of operations, realized by dummy field operations, are known, for example [50].

The most implemented $k\boldsymbol{P}$ algorithm for ECs over $GF(2^n)$ is the algorithm using Lopez-Dahab projective coordinates for the EC point representation [5] that is based on Montgomery's idea [51]. This algorithm is very fast and can be considered as a special version of the *double-and-add-always* algorithm:

Algorithm 6: Montgomery $k\boldsymbol{P}$ algorithm corresponding to [5]

1. $\boldsymbol{Q_0 \leftarrow P;\ Q_1 \leftarrow 2P;}$
2. **for** $i = (l-2)$ **downto** 0
3. ***if*** $k_i = 1$ then $\boldsymbol{Q_0 \leftarrow Q_0+Q_1\ ;\ Q_1 \leftarrow 2Q_1}$
4. ***else*** $\boldsymbol{Q_1 \leftarrow Q_0+Q_1\ ;\ Q_0 \leftarrow 2Q_0}$
5. Output $\boldsymbol{Q_0 = k \cdot P}$

Corresponding to [5] the algorithm is a version of the binary $k\boldsymbol{P}$ algorithm with the following feature: the difference of points $\boldsymbol{Q_1}$ and $\boldsymbol{Q_0}$ is always equal to the input point $\boldsymbol{P}=(x,y)$, i.e.: $\boldsymbol{Q_1 - Q_0 = P}$. In the Montgomery $k\boldsymbol{P}$ algorithm the processing of each bit of the scalar k in the main loop consists of a point doubling and a point addition always, i.e. the Montgomery $k\boldsymbol{P}$ algorithm is highly regular. Due to this fact, the Montgomery ladder is reported in literature [52]-[55] as resistant against simple SCA attacks. Important is the fact, that only in [52] the assumption about the indistinguishability of the writing in different registers is denoted as the important basis of the resistance against simple SCA attacks[4]. In the well-known paper [53] this critical assumption was already forgotten and transformed into the fact that Montgomery ladder, including Montgomery $k\boldsymbol{P}$ algorithm using Lopez-Dahab projective coordinates for EC over $GF(2^n)$ is protected against simple SCA[5]. Other well-known authors [54] write that "a regular execution can thwart these [simple] attacks" and refer [52], without mentioning his main assumption. Similar, in [55] is written that "…Montgomery Ladder are used to prevent SPA/SEMA and timing attacks". Such statements give the impression that $k\boldsymbol{P}$ designs implementing the Montgomery ladder are already successfully protected against simple SCA attacks, due to the regularity of the $k\boldsymbol{P}$ algorithm selected for the implementation.

The next level of the "regularity" strategy for ECs over $GF(p)$ is the "SCA atomicity" principle that was introduced in [38]. It aims at performing all operations, including write-to-register operations, regular to make the power profile of the EC point doubling indistinguishable from the one of the EC point addition. This is especially important for $GF(p)$ implementations, because the dummy EC point addition in a *double-and-add always* algorithm takes a lot of time and energy. Applying dummy field and/or write-to-register operations is a more efficient strategy.

[4] Citation from [52]: "Provided that the writing in registers R0 and R1 (resp. that the squaring of registers R0 and R1) cannot be distinguished from a single side-channel measurement, the Montgomery ladder can be implemented to prevent a given [simple] side-channel attack."

[5] Citation from [53]: "Implementations based on the Montgomery ladder [12]–[14], shown as Alg. 3, are protected against timing attacks and simple SCA since the execution time of the scalar multiplication is inherently unrelated to the Hamming weight of the secret scalar."

There exist many techniques to convert an algorithm to one protected against simple side-channel analysis attacks using the atomicity principle. Best known are the atomic patterns proposed by the following authors:

- B. Chevallier-Mames, M. Ciet and M. Joye [38]
- P. Longa [39]
- C. Giraud and V. Verneuil [40]
- F. Rondepierre [2]

Historically, the first one was proposed in 2004 by [38] and was applied for RSA and ECC (*double-and-add* algorithm). Its main idea is to split the process related to the secret bit (doubling and addition) into the repetition of several sufficiently small *side-channel atomicity blocks.* These blocks represent a sequence of instructions that are indistinguishable – equivalent – from the side-channel analysis point of view. The indistinguishability is achieved by insertion of dummy operations. The doubling and addition operations were represented by 10 and 16 atomicity blocks respectively, each of the blocks consisted of one multiplication *M*, two additions *A* and one negation *N* ("*M-A-N-A*" atom). The total costs for a point doubling operation are 10 multiplication and 20 additions; for a point addition – 16 multiplications and 32 additions. Please note that the side-channel-atomicity is based on the important assumption that the loading/storing of values from different registers are equivalent operations (see section 3.1 in [38]).

In 2007 Longa [39] improved the algorithm proposed in [38] in terms of reducing the number of atomicity blocks and their structure. Based on the "*M-A-N-A*" atom the new calculation sequence for a point doubling requires 8 and a point addition requires 11 such atoms. The cost of point doublings was reduced down to 8 multiplications and 16 additions; the cost of point additions was defined as 11 multiplications and 22 additions. Additionally a new single atomic block was proposed that consists of 2 multiplications, 3 additions and 2 negations ("*M-N-A-M-N-A-A*" atom). 4 such atoms are necessary for a point doubling (i.e. 8 multiplications and 12 additions) and 6 of such atoms for a point addition (i.e. 12 multiplications and 18 additions).

In 2010 Giraud et al [40] presented their implementation of the atomicity principle for the EC scalar multiplication algorithm focused on embedded devices. Their atomic pattern for the *right-to-left* binary mixed coordinates algorithm consists of 2 squaring operations *Sq*, 6 multiplications, 6 additions, and 4 subtractions *Sub* ("*Sq-A-M-A-M-A-M-A-A-Sq-M-A-Sub-M-Sub-Sub-M-Sub*" atom). The point doubling and point addition consist of one and two such big atomic patterns respectively, i.e. a point doubling costs 8 multiplications and 10 additions of field elements and a point addition costs 16 multiplications and 10 additions of field elements.

An atomic pattern consisting of 2 squaring operations, 8 multiplications, 5 additions and 5 subtractions was presented by Franck Rondepierre in 2014 [2]: "*M-Sub-A-M-A-M-M-A-M-M-A-A-Sq-Sub-Sq-Sub-Sub-M-M-Sub*"[6]. The same single atomic pattern is used in the case of point addition, subtraction or doubling, making it the fastest implementation of the algorithm with the atomicity principle: a point doubling as well as a point addition require 10 multiplications and 10 additions of field elements. Additionally, the proposed pattern allows to securely perform double scalar multiplications using the Straus-Shamir trick [56]-[57].

The regularity and atomicity principles are the basic countermeasure principles against simple SCA attacks. But all these countermeasures are based on the assumption that the differences in power profiles of "atoms" are small and depend on the data processed in the "atom".

[6] We implemented this algorithm, see more details in section 8.1

In the literature, for example in [2], it is mentioned that the atomicity algorithms provide resistance against horizontal attacks, applying EC point coordinate randomization [48] many times during a single $k\boldsymbol{P}$ execution.

In our investigation we show that the atomicity algorithms, as well as highly regular Montgomery ladder, are vulnerable to horizontal attacks as well as to our automated simple SCA attacks. The vulnerability is due to the key-dependent addressing of the registers and other blocks in algorithms, i.e. it is an inherent feature of the algorithms. This means that the basic assumption about the indistinguishability of the loading/storing of values from different registers/blocks has to be revised, at least in case of a hardware implementation of the $k\boldsymbol{P}$ algorithm using side-channel atomicity blocks.

2.3.2 **Randomization**

The more the attacker knows about the processed data – the easier it is to reveal the key. Reducing this information can be achieved using randomization techniques. The idea of randomization is somehow the exact opposite of the regularity and atomicity principle. Here it is no longer the goal to ensure that the power shapes of a '0' and a '1' are similar. They may look different, but their distinguishability becomes meaningless to the attacker, i.e. the goal is to ensure that the attacker does no longer know whether a '0' or a '1' are processed in each time slot corresponding to the processing a key bit. This can be achieved by means applied on different levels of abstraction, i.e. randomization of the processed data, the processing sequence, randomizing the underlying hardware and adding noise.

The well-known techniques for randomizing the processed data are blinding, doubling, masking and splitting [48], [58]. These techniques are based on mathematical approaches that allow to process new, randomly chosen and by the attacker unknown input values and, despite that, obtain correct calculation results:

- Randomization of inputs so, that the attacker no longer knows which data are processed increases the effort to determine the key. Randomization of the inputs can be achieved for example using blinding of the EC point $\boldsymbol{P}$ or randomization of the EC point coordinates.

  Point blinding: A random point $\boldsymbol{R}$ has initially to be stored inside the cryptographic chip. Additionally, the kR operation is performed on the chip and the result $\boldsymbol{Q}=k\cdot\boldsymbol{R}$ is also stored. Here k is the private key. For the execution of the $k\boldsymbol{P}$ operation the EC point $\boldsymbol{P}$ is blinded by adding this random point R: $S=P+R$. Instead of executing $k\boldsymbol{P}$, $k\boldsymbol{S}$ is executed. The result is calculated as: $k\cdot\boldsymbol{S}-\boldsymbol{Q}=k\cdot\boldsymbol{P}$. For each new execution of the $k\cdot\boldsymbol{P}$ operation the point $\boldsymbol{R}$ and the point $\boldsymbol{Q}$ have to be refreshed, e.g. by doubling: $\boldsymbol{R'}=2\cdot\boldsymbol{R}$ and $\boldsymbol{Q'}=2\cdot\boldsymbol{Q}$. After the point blinding at the beginning of each $k\boldsymbol{P}$ operation the $k\boldsymbol{S}$ operation with a new point $\boldsymbol{S}=\boldsymbol{P}+\boldsymbol{R'}$ unknown to the attacker will be performed. *Randomization of the projective coordinates of point* $\boldsymbol{P}$. EC point $\boldsymbol{P}=(x,y)$ can be represented using projective coordinates: $\boldsymbol{P}=(X, Y, Z)$, where $x=X/Z$ and $y=Y/Z$. The projective coordinates of a point can be randomized using a random number $\lambda\neq 0$: $(X, Y, Z)=(\lambda X, \lambda Y, \lambda Z)$. The randomization is usually done before each new execution of the $k\boldsymbol{P}$ operation but can be implemented also after each point addition and doubling. After the randomization of the projective coordinates of point $\boldsymbol{P}$ at the beginning of each $k\boldsymbol{P}$ operation the coordinates of point $\boldsymbol{P}$ are unknown by the attacker.

- Randomization of the processed scalar k (private key) before it is processed which means that the traces an attacker records do not reflect the behavior of the implementation when processing the key.

The randomized key k' is calculated as: $k'=k+r\cdot\#\varepsilon$. Here k is the private key and r is a random number smaller than $\#\varepsilon$, which is the number of points of the elliptic curve and is a public parameter of the EC. Thus, $\boldsymbol{Q}=k'\cdot\boldsymbol{P}=(k+r\cdot\#\varepsilon)\cdot\boldsymbol{P}=k\cdot\boldsymbol{P}+\boldsymbol{O}=k\cdot\boldsymbol{P}$. After the key randomization at the beginning of each $k\boldsymbol{P}$ operation the $k'\cdot\boldsymbol{P}$ operation with a new scalar k' that is unknown to the attacker will be performed.

- Randomization of intermediate data, e.g. coordinates of intermediate EC points can be randomized during the $k\boldsymbol{P}$ execution.
- All these approaches can be applied in combinations.

Not only the processed data can be used to randomize the traces recorded by an attacker but the execution of the algorithm can also be used to avoid patterns that allow revealing the key:

- Randomization of the calculation sequence [59]-[61], [62]: operations of the algorithm are executed at different points in time whenever the algorithm is started, in consequence statistical processing to reduce noise becomes more tricky. One of the implementation examples of the sequence randomization of EC point doublings and EC point additions is the Montgomery ladder described in [63].
- The effect of modules that provide the same functionality but have different power consumption will be very similar to randomization techniques as it hinders determining which operation was executed at a certain point in time [64]-[65]. In [66] the authors even propose to change the mapping of operations – here partial multiplications – to hardware units that provide the required operations.
- An asynchronous design might be an additional means to increase entropy in measured traces as the design by default avoids the synchronization that is normally provided by the clock signal. But fully asynchronous designs and attacks against them have not been reported yet. Implementing the Montgomery $k\boldsymbol{P}$ algorithm as an asynchronous design can increase its resistance against SCA significantly but the area and the energy consumption of such implementation will be enormous. The split between these two strategies – the synchronous and the asynchronous designs – can be the Globally Asynchronous Locally Synchronous (GALS) architecture. A GALS circuit consists of a set of modules which are locally synchronous, i.e. each module has its own clock. The frequency of the clock signals should differ for the modules and/or can be implemented with an unstable clock signal frequency for each of the modules. The modules communicate via asynchronous wrappers. A GALS-ification of an IHP $k\boldsymbol{P}$ design is reported in [67].

There are additional means that should be applied when implementing a cipher algorithm:

- Independent of whether the algorithm is implemented in software or hardware, the complexity of attacks can be increased using random interrupts or delay generation techniques, i.e. required operations are executed only after a certain random delay etc. [68]. In particular, the activity of the multiplier as a source of the noise can be combined with delay [69] with the goal to reduce the success of SCA attacks.
- Injection of noise can be a good countermeasure against a broad spectrum of attacks.

The methods randomizing processed data are effective if the attacker analyses the data dependencies in the trace, i.e. in the case of data-bit attacks [36]. For such kind of attacks often at least two measured traces are required for the analysis, i.e. for classical DPA attacks [70] applied to $k\boldsymbol{P}$ traces, correlation power analysis attacks, collision-based attacks [36], [71]-[72] and fault sensitivity analysis [73]. In section 5.1 we show that the randomization methods described above are not sufficiently effective to prevent horizontal address-bit SCA attacks

against EC point multiplication. Please note that also in [37] it was stated that exponent blinding is not an effective countermeasure against horizontal attacks. In section 5.2 we discuss the resistance of a GALS-ified IHP ***kP*** design and show that a straight-forward GALS-ification is not effective against horizontal attacks. In section 5.3 we describe our attack against an open-source implementation of a randomized Montgomery ladder [63]. We attacked simulated power traces and we were able to reveal the processed scalar completely.

Performing of operations in parallel allows to use the activity of "secure" blocks as a kind of non-reducible noise. ECC accelerators implemented in hardware are usually area- and energy-optimized designs. Additionally, a high number of signature verifications/generations per second is often required. Small and low-energy accelerators are slow. Fast accelerators have a big area and high power consumption. The most energy-consuming block is the finite field multiplier. It can be used as a kind of a noise source. If the multiplier is resistant against a broad spectrum of SCA attacks, it can hide the activity of other blocks that are SCA leakage sources. Therefore, an application of a specific multiplication method for the implementation of the finite field multiplier can be a way to reduce the success of both kinds of SCA attacks, i.e. vertical and horizontal attacks. Due to these facts, we concentrated in this work on the use of the multiplier as a kind of countermeasure against SCA attacks.

2.4 Starting point of this thesis

In this section we describe shortly the starting point of my thesis, i.e. the ECC design and the method used for the trace analysis which were available at IHP before this thesis.

The IHP ECC design that was the starting point for this work is denoted as the "basic ***kP*** design" in the rest of this thesis.

2.4.1 The IHP basic *kP* design

The IHP basic ***kP*** design is a hardware implementation of a ***kP*** operation for the EC *B-233*. The EC *B-233* is an EC over binary extended Galois Fields $GF(2^n)$ standardized by NIST [8]. The basic ***kP*** design is described using Very high-speed integrated circuit Hardware Description Language (VHDL).

2.4.1.1 Implemented algorithm

1998 Lopez and Dahab showed that the Montgomery ***kP*** algorithm [51] can be performed using only the *x*-coordinate of the EC point ***P*** [5]. Additionally, they proposed to use special projective coordinates of the EC point ***P*** to avoid the division of elements of binary Galois fields, which is the most complex and often executed operation calculating ***kP*** in affine coordinates. These optimizations reduced the execution time and the energy consumption of the ***kP*** calculation significantly. Algorithm 7 shows one of the most referenced versions of the Montgomery ***kP*** algorithm [6].

The inputs in Algorithm 7 are long binary numbers: the scalar *k* and two affine coordinates of the EC point $\boldsymbol{P}=(x,y)$. The outputs are the affine coordinates of the EC point $(x_1,y_1)=k\boldsymbol{P}$. Additionally, 5 variables X_1, X_2, Z_1, Z_2 and T are used in the algorithm for data storing. The variables represent finite field elements.

Algorithm 7 is a bitwise processing of the scalar *k*. It can be observed as a sequence of mathematical operations – multiplications, squaring operations and additions – of elements of a finite field. Intermediate and end results have to be written to registers. Line 1 is the so-called initialisation phase of the algorithm; it is the conversion of the affine coordinates (x,y) of the input point ***P*** into projective Lopez-Dahab coordinates: $(x,y)\rightarrow(X,Y,Z)$. The initialisation phase

can be observed also as a sequence of operations for the processing of the most significant bit of the scalar k, whereby it is assumed that the most significant bit is not zero: $k_{l-1}=1$. Lines 2-10 are the main loop of the algorithm. Each loop corresponds to the processing of a single bit value of the scalar k. Lines 11-12 represent the calculation of the affine coordinates of the $k\boldsymbol{P}$ result. Here are two divisions of the finite field elements. Please note that the division is the most complex finite field operation. Using the projective Lopez-Dahab coordinates the division has to be calculated only for the conversion of the projective coordinates into affine ones, but no division is calculated in the main loop of the algorithm. The next complex operation is the multiplication of finite field elements.

Algorithm 7: Montgomery kP using projective Lopez-Dahab coordinates

Input: $\boldsymbol{k} = (\boldsymbol{k_{l-1} \ldots k_1 k_0})_2$ with $\boldsymbol{k_{l-1}=1}$, $\boldsymbol{P=(x,y)}$ *is a point of EC over* $GF(2^l)$
Output: $\boldsymbol{kP = (x_1, y_1)}$

1: $X_1 \leftarrow x,\ Z_1 \leftarrow 1,\ X_2 \leftarrow x^4+b,\ Z_2 \leftarrow x^2$
2: ***for*** $i=l-2$ ***down to*** 0 ***do***
3: ***if*** $k_j=1$
4: $T \leftarrow Z_1,\ Z_1 \leftarrow (X_1Z_2+X_2T)^2,\ X_1 \leftarrow xZ_1+X_1X_2TZ_2$
5: $T \leftarrow X_2,\ X_2 \leftarrow T^4+bZ_2^4,\ Z_2 \leftarrow T^2Z_2^2$
6: ***else***
7: $T \leftarrow Z_2,\ Z_2 \leftarrow (X_2Z_1+X_1T)^2,\ X_2 \leftarrow xZ_2+X_1X_2TZ_1$
8: $T \leftarrow X_1,\ X_1 \leftarrow T^4+bZ_1^4,\ Z_1 \leftarrow T^2Z_1^2$
9: ***end if***
10: ***end for***
11: $x_1 \leftarrow X_1/Z_1$
12: $y_1 \leftarrow y+(x+x_1)[(X_1+xZ_1)(X_2+xZ_2)+(x^2+y)(Z_1Z_2)] \,/\, (xZ_1Z_2)$
13: **return** (x_1, y_1)

The sequence of operations in Algorithm 7 can be realized in different ways. Depending on the requirements, e.g. small area or short execution times, different optimizations can be applied. For example, if an area optimization is the aim, the number of registers can be reduced, only a single multiplier has to be implemented and/or the multiplication can be implemented serially. With the goal to accelerate the $k\boldsymbol{P}$ execution some operations in Algorithm 7 can be executed in parallel. It reduces the execution time of the $k\boldsymbol{P}$ operation and increases the resistance of the ECC design against SCA attacks at the same time.

The IHP basic $k\boldsymbol{P}$ design was a kind of a security-aware, area-, energy- and time-optimized implementation of the EC point multiplication based on the Algorithm 7, i.e. Algorithm 7 was slightly modified to reach a trade-off of the requirements. This modification allows to implement all operations in parallel to the field multiplications; that increases the efficiency of the design and provides some kind of robustness against power analysis attacks. Algorithm 8 shows the modified Montgomery $k\boldsymbol{P}$ algorithm.

The main modifications compared to Algorithm 7 are:

- A simplified sequence in the initialization phase and the processing of the second most significant bit of the scalar k before the main loop of the algorithm.

  The initialization of register Z_1 with the value 1 (see line 1 in Algorithm 7: $Z_1 \leftarrow 1$) can be exploited for successfully revealing the value of k_{l-2} using simple power analysis of the corresponding part of the power trace. Besides this, the power profile of the processing of k_{l-2} can be used for understanding implementation details of the analysed design and thus for preparing templates for further attacks. The processing of k_{l-2} before the main loop using a new sequence of operations for the initialization as well for processing k_{l-2} is a kind of countermeasure again this vulnerability. Another way to countermeasure this vulnerability is for example the randomization [48] of the projective coordinates of the EC point $\boldsymbol{P}$.

- Adaptable sequence of the multiplications $X_1 \cdot Z_2$ and $X_2 \cdot Z_1$ in line 11 (both multiplications are marked blue in Algorithm 8[7])

  A straight-forward implementation of Algorithm 7 means that the value of the currently processed key bit k_i defines the sequence of the multiplications in the main loop of the algorithm: the 1st multiplication is $X_1 \cdot Z_2$ if the current key bit is '1' (see line 4 in Algorithm 7); otherwise the 1st multiplication is $X_2 \cdot Z_1$ (see line 7 in Algorithm 7). In the IHP basic ***kP*** design the sequence of these two multiplications for the processing of the current key bit value k_i depends on the previously processed key bit value[8] k_{i+1}. This means, that for k_i =1 either $X_1 \cdot Z_2$ or $X_2 \cdot Z_1$ should be performed first, depending only on the value of k_{i+1}. More details can be found in [74]-[75]. [76] also reports on an implementation of the 1st and 2nd multiplication as an adaptable sequence.

- The division is calculated only once, compare lines 11-12 in Algorithm 7 and lines 18-20 in Algorithm 8.

  The division is implemented as a sequence of field squaring operations and field multiplications using the little theorem of Fermat[9].

Algorithm 8: Modified Montgomery algorithm for the *kP* operation

Input: $k = (k_{l-1} \ldots k_1 k_0)_2$ with $k_{l-1} = 1$, $P=(x,y)$ *is a point of EC over* $GF(2^l)$
Output: $kP = (x_1, y_1)$

1: $X_1 \leftarrow x$, $X_2 \leftarrow x^4 + b$, $Z_2 \leftarrow x^2$ *//initialization*
2: **if** $k_{l-2} = 1$ **then** *//processing second most significant bit*
3: $T \leftarrow Z_2$, $Z_1 \leftarrow (X_1 Z_2 + X_2)^2$,
$X_1 \leftarrow X_1 Z_2 X_2 + x Z_1$,
$U \leftarrow b\, Z_2^4$, $X_2 \leftarrow X_2^4 + U$,
$U \leftarrow T Z_2$, $Z_2 \leftarrow U^2$.
5: ***else***
6: $T \leftarrow Z_2$, $Z_2 \leftarrow (X_1 Z_2 + X_2)^2$,
$X_2 \leftarrow X_1 X_2 T + x Z_2$,
7: $T \leftarrow X_1$, $U \leftarrow b X_2^4$, $X_1 \leftarrow X_1^4 + b$,
$U \leftarrow T X_2$, $Z_1 \leftarrow T^2$.
8: ***end if***
9: ***for*** *i* ***from*** *l* − 3 ***downto*** 0 ***do*** *//start of the main loop*

10: ***if*** $k_i = 1$ ***then***
11: $T \leftarrow Z_1$, $Z_1 \leftarrow (\mathbf{X_1 Z_2} + \mathbf{X_2 Z_1})^2$, $X_1 \leftarrow x Z_1 + X_1 X_2 T Z_2$,
12: $T \leftarrow X_2$, $X_2 \leftarrow X_2^4 + b Z_2^4$, $Z_2 \leftarrow T^2 Z_2^2$.
13: ***else***
14: $T \leftarrow Z_2$, $Z_2 \leftarrow (X_2 Z_1 + X_1 Z_2)^2$, $X_2 \leftarrow x Z_2 + X_1 X_2 T Z_1$,
15: $T \leftarrow X_1$, $X_1 \leftarrow X_1^4 + b\, Z_1^4$, $Z_1 \leftarrow T^2 Z_1^2$.
16: ***end if***
17: ***end for*** *//end of the main loop*
//calculating affine coordinates of the kP result
18: $x_1 \leftarrow 1 / (x Z_1 Z_2)$
19: $y_1 \leftarrow y + (x + x_1)[(X_1 + x Z_1)(X_2 + x Z_2) + (x^2 + y)(Z_1 Z_2)] \cdot x_1$
20: $x_1 \leftarrow X_1 x_1 x Z_2$ *// i.e.* $x_1 = X_1 / Z_1$
21: **return** (x_1, y_1)

[7] With the goal to simplify the explanation I do not show the corresponding extension of the algorithm as pseudo-codes.
[8] Please note that the Montgomery-based ***kP*** algorithms process the scalar *k* starting from its most significant bit down to its least significant bit, i.e for the currently processed bit k_i the previously processed bit is k_{i+1}.
[9] The little Fermat's theorem is: $x^{2^l} \bmod f(t) \equiv x \Rightarrow x^{2^l - 2} \bmod f(t) \equiv x^{-1}$

2.4.1.2 Structure of the IHP basic kP design

Fig. 1 shows the structure of the IHP basic ***kP*** -accelerator.

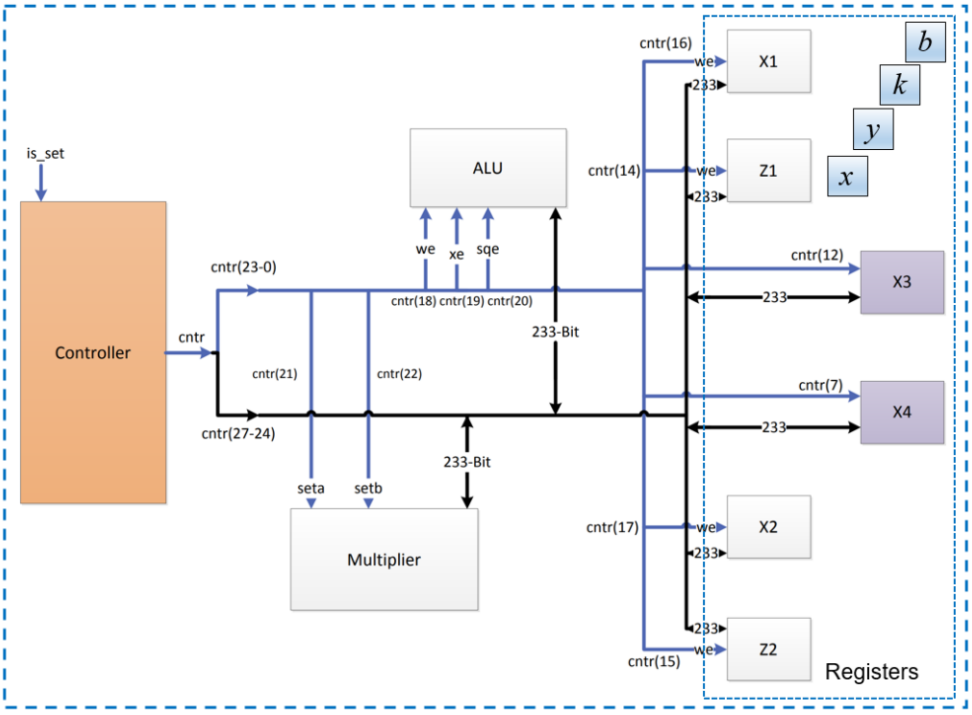


Fig. 1. Structure of the IHP basic ***kP*** accelerator, based on [74].

The ***kP*** accelerator obtains a scalar k and two affine coordinates x and y of a point ***P*** of the EC *B-233* as inputs. The numbers x, y and k are up to 233-bit long binary numbers that represent elements of $GF(2^{233})$, with the irreducible polynomial $f(t)=t^{233}+t^{74}+1$. The parameter b of the equation of the EC is pre-stored in a register.
The design consists of four blocks: Controller, field Multiplier, ALU and Registers. The block Controller manages the sequence of the field operations. It controls the data flow between the other blocks and defines which operation has to be performed in the current clock cycle. Depending on the signals of the Controller the block ALU performs addition or squaring of its operands. The design consists of only one block Multiplier that calculates the field product of 233-bit long operands using 9 partial products only according to a fixed calculation plan. The design contains ten 233-bit long registers for saving input and output as well as intermediate values. The registers X_1, X_2, Z_1, Z_2, X_3 and X_4 are necessary for the implementation of the main loop of the algorithm. The input registers x and y are used also as output registers, i.e. they contain the result of the EC point multiplication ***kP*** after its calculation.

The field multiplier is implemented using the 4-segment Karatsuba multiplication method [77]. The field multiplier uses a partial multiplier for 59-bit long operands.

The data exchange between the blocks/registers is realized using a muxer called *bus* in the rest of this thesis. It consists of many logic gates that react on the address given by the Controller. The *write-to-bus* operation connects the output of the addressed block to the inputs of all other blocks. By the *read-from-bus* operation only the addressed block accepts the values on its input as data for processing.

All blocks of the IHP basic ***kP*** accelerator were designed in years 2001–2006. The field multiplication method is patented by IHP [78]. The basic ***kP*** accelerator described in [74] is

a redesigned version of an earlier implemented IHP ***kP*** design: it is an efficient and secure-aware version. The 6 field multiplications, 5 field squaring operations and 3 field additions necessary to process a bit of the scalar k in the main loop are executed in parallel, whereby the sequence of these operations is independent of the processed bit value. Due to the fact that the new multiplicands are written into the field multiplier in parallel to the calculation of the current field product, each field multiplication requires 9 clock cycles only to be calculated. Thus, the ***kP*** accelerator needs only 6·9=54 clock cycles for the processing a key bit in the main loop.

2.4.2 **Analysis tools and methods**

In [74]-[75] and [79] a method for a statistical analysis of simulated power traces was used.

The method was realised in IHP by means of the TAMPRES-Project in 2013 [80] and exploits the null hypothesis. Each single measured or simulated ***kP*** trace is a bitwise processing of the secret scalar k. The scalar is a sequence of '1' and '0' bit values. Thus, the trace measured during a ***kP*** execution or simulated for a ***kP*** execution (further ***kP*** trace) is a set of two populations: '1'-shapes and '0'-shapes.

The ideal case from the designer's point of view is that the both populations are not distinguishable from each other, i.e. their mean shapes $\boldsymbol{m_0}$ and $\boldsymbol{m_1}$ are pointwise (or sample-wise) equal. In case if the populations are distinguishable their mean shapes are different, i.e. $\boldsymbol{m_0 \neq m_1}$. This means that the mean shape of the both populations $\boldsymbol{m}$ is between the both shapes: $\boldsymbol{m_0 < m < m_1}$ or $\boldsymbol{m_0 > m > m_1}$**.** Based on this assumption and even without any knowledge about the number of samples in each population an attacker can try to separate the ***kP*** trace, i.e. the set of two populations – '0'-shapes and '1'-shapes – into two subsets.

This idea was implemented in IHP in Microsoft Excel using programmed tabs. This tool allowed to analyse a ***kP*** trace simulated with the time simulation step equal to the clock cycle period only, i.e. each slot corresponding to the processing a key bit value contained only 54 samples (values). Using this Excel-tool 54 key candidates were extracted and compared to the real scalar k that was used in the power simulation.

To evaluate the success of the attack for each key candidate its *relative correctness* δ was calculated. This is the number of the correctly revealed bits in the key candidate divided by all revealed bits represented in %. The relative correctness δ is a value between 0% and 100%. If a key candidate has the correctness δ=100% it means that the key candidate is equal to the real key k.

This analysis method is effective if the '1'-shapes differ from the '0'-shapes and if both populations consist of almost the same number of samples.

This analysis method can be observed as a kind of a difference of means that an attacker can perform without any knowledge about the scalar k. Due to this, it was denoted in publications [74]-[75] and [79] as "difference of means". This analysis method as an Excel tool was one of the starting points of this thesis. In this work it was improved, accelerated, implemented in Java as a part of IHP SCA Tool (see section 3.2.3) and denoted in later publications as the *comparison to the mean* analysis method, see for example [81].

2.4.3 **Resistance of the basic *kP* design against attacks**

A fast and low-cost horizontal attack, for example the one described in the previous subsection, is a good means to determine the clock cycles of the main loop implementation with a strong SCA leakage source. A high relative correctness of extracted key candidates, i.e. a correctness close to 100% or to 0%, shows the success of the performed attack as well as the

clock cycles with strong SCA leakage. Especially this aspect is important for designers because it can be helpful for localizing process(es) causing the SCA leakage. Please note that the worst-case of the attack result from the attacker's point of view is a correctness of 50 % for all extracted key candidates which means that the comparison to the mean method cannot even provide a slight hint whether the key bit processed is more likely a "1" or a "0", i.e. this means that the attack was not successful at all. The worst-case from the attacker's point of view is the ideal case from the designer's point of view.

The IHP analysis method described above was applied in [74]-[75] for the statistical analysis of an old IHP ECC design. Many key candidates with a correctness of 100 % were extracted. After the analysis of the processes causing this high success rate some vulnerabilities were determined and the ***kP*** implementation was redesigned, i.e. a new sequence of operations within the main loop was proposed. The redesigned version, i.e. the IHP basic ***kP*** design, was implemented and detailed described in [75]. The resistance of the basic ***kP*** design against the same horizontal attack was evaluated using the same IHP analysis method. For the evaluation, the IHP basic design was synthesised for the IHP 130 nm technology and power traces of two ***kP*** executions were simulated using the Synopsys PrimeTime suite, version from 2014 [82]. Both ***kP*** executions processed the same point ***P*** but different scalars *k1* and *k2*. The simulations were performed not only for the top design but also for all individual design blocks. Thus, for both ***kP*** executions the power traces of the whole design and for the selected design blocks (field multiplier, ALU and registers X_1, X_2, Z_1, Z_2, X_3, X_4) were statistically analysed. Attacks using the power trace of the whole design were not successful: only few key candidates with a correctness of 70 % were extracted. The correctness of other key candidates was between 40% and 60 %. By analysing the power traces of the ALU and registers many key candidates with a correctness of 100% were extracted. But due to the fact that the success of the attack analysing the power trace of the whole ***kP*** design was not high, no additional investigations were done. The ideas for re-design applied in [75] for the basic ***kP*** design were verified on the other ***kP*** design for the same EC *B-233* using a faster field multiplier. For this new design 36 key candidates were extracted. The correctness of the best key candidate was 76%.

The goals of the re-design performed in [75] were the SCA resistance (i.e. using the regularity principle) and time optimization of Montgomery ***kP*** algorithms for the EC *B-233*. The field multiplier defines the minimal possible execution time of the operations in the main loop, in case of the basic ***kP*** design there are 6 multiplications, each 9 clock cycles long. Thus, 6x9=54 clock cycles can be achieved as the minimum execution time for each main loop iteration if all other operations are performed in parallel to the multiplication. All field additions, field squaring operations as well as the register operations were implemented in parallel to the field multiplications in [75], i.e. it was an efficient implementation. The field multiplier is the most energy-consuming unit, i.e. its activity is a kind of noise if the activity of other blocks, for example the registers, is analysed. In [79] it was shown on example of the IHP 130 nm technology that efficient implementations of the Montgomery ***kP*** algorithm performing operations in parallel to the field multiplications are inherently more resistant against horizontal attacks than inefficient implementations. The registers and the block ALU were identified as strong SCA leakage sources in [79].

To summarize, the starting point of this thesis is the IHP basic ***kP*** design, that is an efficient implementation of the Montgomery ***kP*** algorithm for the EC *B-233*, without any additional countermeasures against physical attacks.

3 Collection, preparation and analysis of traces

The success of physical attacks analysing measured traces depends at first hand on the quality of measurements, i.e. it depends on the used measurement equipment as well as on the correctness of its applying. For example, the used magnet field probe, its position and orientation can influence the signal/noise ratio of measured electromagnetic traces, significantly. The next most important factor is the sampling rate applied by measurements or simulations [83]. Trace preparation, statistical methods used for the analysis, etc. influence the success of the attacks as well. In this section we explain all the details regarding trace capturing as well the implemented analysis methods and criteria for evaluation of the attack success that we applied in this thesis.

3.1 Simulations and measurements of traces

The collection and preparation of $k\boldsymbol{P}$ traces for statistical analysis are important steps when evaluating the SCA vulnerability of cryptographic designs. Generally, simulated power traces of $k\boldsymbol{P}$ executions as well as measured power and electromagnetic traces can be statistically analysed to evaluate the resistance of $k\boldsymbol{P}$ designs to SCA attacks. Horizontal attacks are a good means for evaluating the resistance $k\boldsymbol{P}$ accelerators.

To the best of our knowledge, the simulation of 3D electromagnetic radiation of big circuits is a really complex task. Commercial tools solving this task are not available yet. Only some custom simulators performing electromagnetic cartography for a localized electromagnetic analysis attack against an AES design are reported in the literature [84]-[86]. Thus, in early phases of developing designs, i.e. before manufacturing of ASICs, only power traces can be simulated and analysed. The use of simulated power traces saves the manufacturing costs and allows to analyse the power consumption of single design blocks. Thus, the design blocks that are the strong SCA leakage sources can be determined and this knowledge can be used for a re-design.

In this section we explain how we simulated and prepared the power traces for the SCA analysis. The second aspect covered in this chapter is how power and electromagnetic emanation traces are captured.

3.1.1 Simulation of power traces

The basic ECC design available at IHP as well as other ECC accelerators designed by me were implemented in hardware using VHDL. We synthesized the designs using the IHP 130 nm as well as the IHP 250 nm technology libraries and simulated the power traces (PTs) of the $k\boldsymbol{P}$ execution using the Synopsys PrimePower suite for randomly generated inputs – the EC point $\boldsymbol{P} = (x, y)$ and a 232-bit long private key k. The processed values are given below in hexadecimal[10]:

x=181856ADC1E7DF1378491FA736F2D02E8ACF1B9425EB2B061FF0E9E8246
y=9FED47B796480499CBAA86D8EB39457C49D5BF345A0757E46E2582DE6
k=93919255FD4359F4C2B67DEA456EF70A545A9C44D46F7F409F96CB52CC

The PrimePower tool uses the gate-level simulation activity stored in the activity file to estimate the averaged power consumption of the design. It can also precisely analyse the power consumption of the design for each clock cycle within a given time interval. The time

[10] These are the values mostly used in my experiments. We also experimented with different scalars and EC point coordinates. All the values are given in section 5.1.

interval can be set down to 10 ps, i.e. the simulation of the power trace(s) can be done with a very high time resolution.

During this time interval, the energy of the gates switching will be accumulated and represented as the mean power within this time interval, i.e. it is a kind of compression of the power trace. If the determined time interval is long, for example equal to the period of a clock signal, the clock cycle will be represented using only 1 simulated sample. The value of the sample will be equal to the mean power of the clock cycle.

By measurements, opposite to simulations, each trace consists of only the measured samples. Any information about energy consumed between the measured values is not accumulated by measurements and therefore is lost. The number of measured samples is determined by the sampling rate of the oscilloscope applied for the measurements. Oversampling can result in a long analysis time, big memory requirements, etc. Undersampling can reduce the success rate of attacks, due to the lost information about the current drawn from the power supply or the electromagnetic emanation during a $k\boldsymbol{P}$ execution. Thus, the undersampling is dangerous if the resistance of cryptographic implementations has to be evaluated by analysing measured traces.

In simulated traces there is no loss of information and additionally, such traces are noiseless. Due to these facts, we set in many (but not in all) of my experiments the time interval for the simulation equal to the clock cycle period, i.e. we represented each clock cycle using only one power value – the average power value of the clock cycle. This is reasonable, as it simplifies the statistical analysis of the trace without any loss of information relevant for the analysis. We performed some experiments using a very high time resolution, i.e. we set the time interval for the simulation to 0.01 ns. With a period of 30 ns of the clock signal that results in 300 samples per clock cycle. We used such traces to investigate the influence of different compression methods – not only averaging – on the success of the performed SCA attacks. Additionally, we used uncompressed traces for implementing and performing the selected horizontal SCA, for example for automated simple SCA (see section 3.2.5.1) and for extraction of the processed scalar k using Fourier transformation (see section 3.2.5.2).

Fig. 2 shows schematically the design synthesis flow at IHP, including power analysis steps.

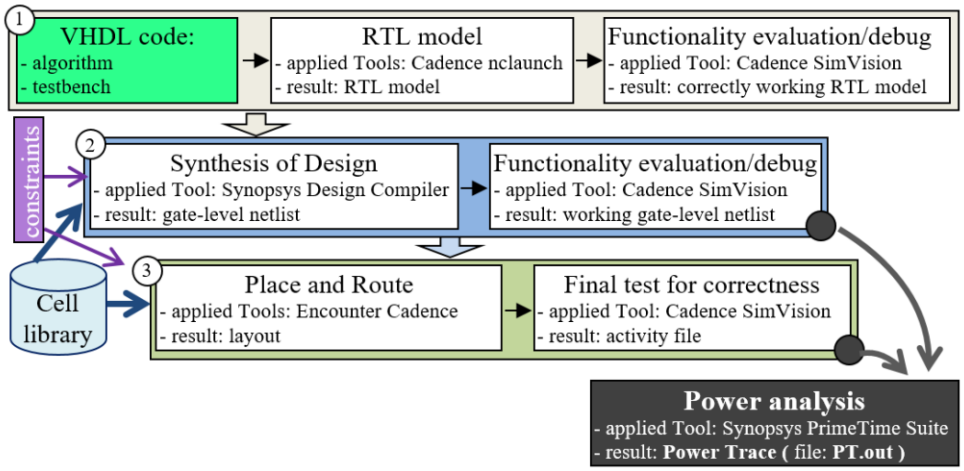


Fig. 2. ASIC design synthesis flow at IHP including power analysis steps.

In order to implement an algorithm in hardware, it has to be described using for example the Very high speed integrated circuits Hardware Description Language (VHDL). The VHDL code is the input for the synthesis tools (see step 1 in Fig. 2). After the functionality of an algorithm is described, a design synthesis tool has to be applied to create a simulator for the

hardware design. It can be for example a Verilog Compiler Simulator Tool suite from Synopsys or Cadence nclaunch [87] , which is available at IHP. In this step a model of the flow of the signals between registers and logic gates is implemented, i.e. a register-transfer level (RTL) model is realized. To be tested, the design needs some inputs and user-specified constraints, for example the target frequency of the clock cycle period. They are stored in a separate file called "testbench". The simulator created using the Cadence software (nclaunch) provides the signal transition information stored as a separate text file denoted as Value Change Dump format (.vcd). This file can be visualized using the Cadence SimVision tools for the visualization of waves. Thus, the RTL model can be used to determine the execution time in clock cycles and to verify and/or debug the functionality of the design.

After the RTL model is working correctly, the design can be synthesized (see step 2 in Fig. 2). For the synthesis of designs detailed information about the used technology is required. This is a description of standard gates, i.e. their functionality and physical parameters such as size, area, signal delay, dynamic power consumption, leakage, etc. We denote this information as "cell library". The Synopsys Design Compiler requires the debugged VHDL code, timing constraints and the used cell library to output a gate-level netlist. After that the correctness of the functionality has to be evaluated. For this test not only a testbench with inputs and user-defined constraints is necessary but also design rule constraints have to be taken into account. An example of design rule constraints is the transition time of certain pins that are specified in the used cell library. Such requirements are denoted as "constraints" in Fig. 2. Please note that all these data, i.e. the algorithm selected for the implementation, its description as VHDL code, the testbench and the user-specified constraints are created or defined by the designers. So far the resistance of cryptographic designs depends directly on the knowledge of the designers regarding different $k\boldsymbol{P}$ algorithms, diversity of attacks against ECC and possible countermeasures.

Step 3 "Place and Route" (layout) in Fig. 2 corresponds to the placement and routing processes in the design flow, i.e. here each gate is placed and then gates are wired (using Cadence tool Encounter/Virtuoso at IHP). Related gates can be grouped together with the goal to reduce routing complexity or/and wire delay using heuristic algorithms. Designers can influence the resistance of the implemented design against SCA attacks "at the gate level". Some critical parts of the design can be implemented manually, without any optimization, for example by using "dummy" gates. For example flip-flops can be placed manually in order to increase the resistance of designs against fault-injection or electromagnetic analysis attacks, etc. After placement and routing the correctness of the design's functionality has to be tested again.

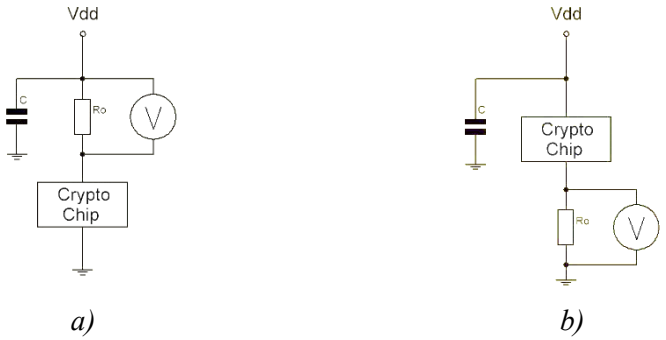
Power Traces (PT) can be simulated using Synopsys' PrimeTime static timing analysis tool [82] after step 3 as well as after step 2 according to the flow, i.e. after or before the "place and route". We denote the result as "PT.out"-file in Fig. 2. The file consists of a header, a list of design blocks and a part with the power consumption of the blocks. The header contains the general information about tools used to generate the power trace (PrimeTime version), time resolution. The list of design blocks contains the description of each block in a separate line. Each block is numbered. The biggest part of the "PT.out"-file is the information about the power of each (numbered) block of the design at any time during the simulation. If the power of a block during a time interval t_i did not change, i.e. it is the same as for the previous time interval t_{i-1}, there is no information shown for this block at the current time t_{i-1}. This data format differs significantly from the measured data, where each line contains a time for the measurement starting from the triggered 0-time point and the measured magnitude value.

To apply statistical analysis to simulated traces we converted the simulated power values for the selected block(s) to a format similar to that of measured data. We extracted the information for each single block from the "PT.out"-file and stored it as a separate file. We automated this process and integrated it into my tool described in section 3.2.6.

Please note that the information about the physical parameters of the gates, that is stored in the standard cell library, is usually evaluated via measurements conducted on special test structures produced for such purposes. Another possibility to evaluate the parameters of the cells is based on the modelling using the averaged measurements performed on a fair amount of single transistors. Surely, the simulated trace depends not only on the quality of the applied simulation tools, i.e. Synopsys PrimeTime, but also on the quality of the measurements for the parametrized gates. But it was expected, that the power traces of a synthesized design, simulated using its layout, will look similar to the traces measured after the design was manufactured (this fact was also evaluated in this thesis). We experimented with power traces simulated before the "place and route" step as well as after it. These details are given in the description of each experiment conducted.

3.1.2 **Measurements setup for power analysis**

The simplest way to measure a power trace is to place a shunt resistor R_0, either in the ground (see Fig. 3-*a*) or power supply line of the circuit (see Fig. 3-*b*). In this case, according to Ohm's law, a drop of voltage proportional to the current is measured on the shunt. However, the voltage drop will affect the circuit as well, which means the value of a shunt has to be selected carefully.



a) *b)*

Fig. 3. Measurements of the Power Traces: a probe can be placed in the power supply line Vdd of the chip (a) or ground line (b).

Another, relatively expensive way, is to apply a differential probe, which utilizes a differential amplifier. Its output voltage is proportional to the difference between two input signals, i.e. the voltage difference between two points in a circuit.
The probes most commonly used in the side-channel analysis are current probes based on current transformers, which inductively measure the current through a conductor. The current transformer is used as a sensor in which the investigated current is flowing through the primary winding and induces proportional voltage in the secondary winding. Such type of probes was used in this work, and they are detailed in the next subsections.

The placement of the current probe also plays an important role (see Fig. 3). It is more preferable to insert the current probe in the power supply line (see Fig. 3-*a*), due to the following reasons. Nowadays, FPGAs, as well as ASICs, may have different separated power domains inside of the chip. For example, a typical FPGA from Xilinx has at least Vcc_int which provides a power supply voltage for the core logic, Vcc_o that powers I/O pins, the auxiliary voltage Vcc_aux etc. From the SCA point of view the Vcc_int voltage is the most interesting one, as it is responsible for the gates switching. Therefore, in contrast to the ground line, placing the probe in the Vcc_int power supply line eliminates all currents drawn from the power supply of other domains from the measurements.

It is obvious that the placement of the shunt/probe is not supposed by the manufacturers of the devices. Therefore, in most cases, the attacked device requires modifications in order to connect/install the probe, which is the main disadvantage of this approach.

At IHP there are two current probes suitable for SCA attacks: a CT-1 current probe (see Fig. 4-*a*) manufactured by Tektronix [88] and a current probe manufactured by Riscure B.V. [89] (see Fig. 4-*b*).

a) *b)*

Fig. 4. Current probes available at IHP: *a)* - Tektronix CT-1 current probe; *b)* - Riscure current probe, amplifier and a power supply.

The CT-1 current probe utilizes the measurement principle of a current transformer and has a bandwidth of 25 kHz to 1 GHz. Due to its bidirectional compatibility, it can be used either for the measurement or injection of a signal. The main parameters according to the datasheet are signal rise time of 350 ns, a sensitivity of 5 mV/mA when terminated at 50 Ohms. Important is that the given sensitivity is related to the wire with the current passing one time through the transformer core and can be increased respectively passing the wire multiple times through the core. The probe has an accuracy equal to ±3% and insertion impedance of 2 Ohms at 100 MHz. The rest of the parameters can be found in the datasheet.

The Riscure current probe is based on the Tektronix CT1 current probe. The probe bandwidth is defined as a value from 1 MHz to 1 GHz as it has an integrated filter. The probe has a sensitivity of 25 mV/mA due to the 5 windings passing through the core of the CT-1 current transformer. To increase the sensitivity of the probe even more, a low-noise amplifier is provided together with a 12 V power supply unit. The manufacturer claims that this set is capable to measure pA current variations [89]. The supplied HD24248 [90] amplifier has an operating frequency range from 100 kHz to 2500 MHz, a typical gain and noise figure of 25 dB and 2.4 dB respectively at 1000 MHz frequency and +25 °C. The rest of the parameters are given in the corresponding datasheet [89]. In comparison to the Tektronics CT-1 probe, a current measured using the Riscure current probe is amplified.

Being a single big company on the market which specializes in the production of side-channel analysis equipment, Riscure positions itself as a market leader in this field. Therefore, its equipment is widely used in academia and security test laboratories. For this reason, to present results that can be compared with other researches we used the current probe from Riscure in my investigations.

3.1.3 **Measurements setup for electromagnetic analysis**

The probe selected for the measurements of electromagnetic radiation, its position and orientation can influence the success of SCA attacks significantly. Using for example an unsuitable electromagnetic (EM) probe or applying even the best suitable probe at a wrong

measurement position/orientation may lead to the fact that there is no leakage detected, whereas an attacker will find a more suitable position for doing measurements for example due to his/her longer experience in attacking devices. In this chapter we describe the setup that we applied in my experiments for collecting electromagnetic traces of $k\boldsymbol{P}$ executions.

3.1.3.1 Measuring electromagnetic radiation: basics

The basic assumption of side-channel analysis attacks is that the current through a cryptographic chip depends on the processed inputs and the used private key. Thus, the current through a cryptographic device can be analyzed to extract the key (Power Analysis attacks). Because the changes of the current through a wire cause the changes of its magnetic field, the magnetic field of the current depends also on the processed inputs and private key and can therefore be analyzed to extract the key (Electromagnetic Analysis attacks). Important is, that in this case not the magnetic field but the rate of its changes will be measured using coils. Nevertheless, the result of the measurements depends extremely on the size, position and orientation of the coil. In this section we explain how the placement and orientation of electromagnetic (EM) probes influence the measurement results on an example of the EM field of a single wire with a flowing current.

Let us assume, there is a long, thin and straight wire with direct current in vacuum (or in the air). Then the current causes a magnetic field that can be characterized using magnetic induction. The magnetic induction (or flux density) $\vec{B}$ is a vector. The magnitude of the magnetic induction at the distance l from the wire can be calculated using the formula:

$$B = \frac{\mu_0 I}{2\pi l} \tag{5}$$

where μ_0 is a coefficient called the permeability of the vacuum. The direction of the vector $\vec{B}$ can be defined with the right-hand rule. The flux of the magnetic field through a surface depends on its area and orientation to the vector $\vec{B}$. Magnetic flux through a surface with the area S is the sum of the normal component of all vectors $\vec{B}$ through this surface:

$$\Phi = \int_S B_{norm}\, dS \tag{6}$$

If the current through the wire is alternating, its magnetic field varies over time. In a coil with area S a voltage will be induced (Faraday's law)[11]:

$$u(t) = -\frac{d\Phi}{dt} = \frac{d(\int_S B_{norm}(t)\, dS)}{dt} \tag{7}$$

If the coil has n turns, the induced voltage is:

$$u(t) = \sum_{i=1}^{n} u_i(t) \tag{8}$$

Accordingly to formulae (6) and (8) coils with different diameters and number of turns can be used to measure the changes of the magnetic field of a wire with a current: the faster the current changes and the closer the coil is to the wire, the higher the measured voltage is. In addition, the orientation of the coil influences the measurement significantly.

Fig. 5 shows some coils at different positions. Coils at positions *a*) and *b*) are placed horizontally. Coils at positions *c*) and *d*) are placed vertically. The positions and orientation *b*) and *d*) of the probe shown in Fig. 5 are not suitable for electromagnetic measurements. The

[11] Static magnetic fields cannot be detected in this way.

coil at position *b*) is placed directly below the wire. For this case all normal components of the vectors $\vec{B}$ through the coil are compensating each other. The magnetic flux will be zero independently of the area of the surface and of the distance to the wire. The coil at position *d*) is oriented parallel to vectors $\vec{B}$, i.e. all vectors $\vec{B}$ are tangential to the surface, i.e. the magnetic flux through the coil will also be zero independently of its area and of its distance to the wire. The most successful positions and orientations of the coil for measurements are positions *a*) and *c*):

- the measurement coils are placed as close as possible to the wire;
- vectors $\vec{B}$ are normal to the inner surface of the coils.

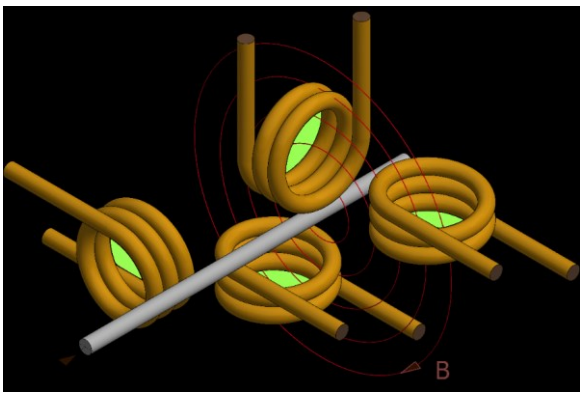


Fig. 5. Examples of the placement and orientation of the coils: positions *a)* and *c)* are best suitable; positions *b)* and *d)* are not suitable for the measurements.

Fig. 6 depicts different placements and orientations of the measurement coils over a chip die.

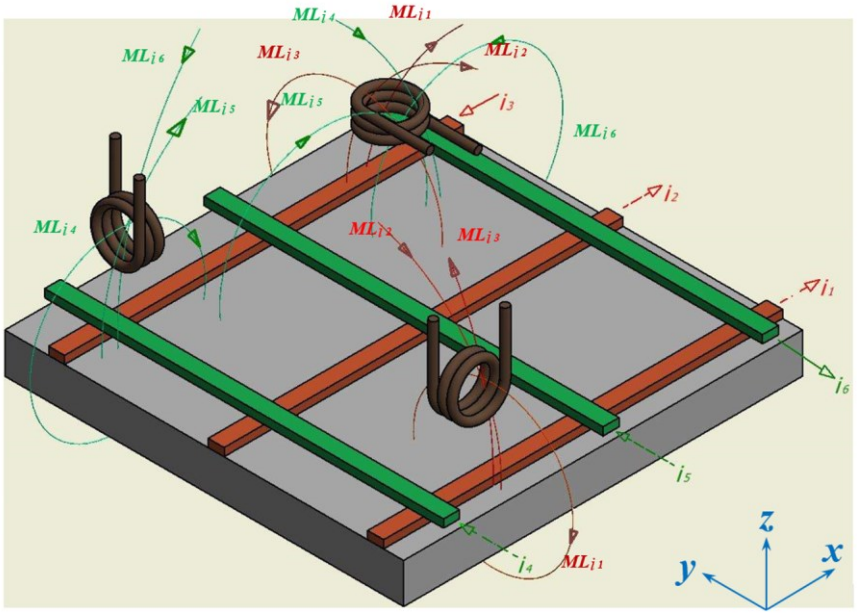


Fig. 6. Differently placed coils over a chip die, schematic representation: the magnetic flux through a surface (coil) depends on all wires with a current.

In Fig. 6 three green wires are placed in the top metal layer. They are thin, straight and directed along the *y*-axis. Three red wires are placed in the next lower metal layer. They are thin, straight and directed along the *x*-axis. The red wires are orthogonal to the green ones. Let's assume, that we can fix a certain moment in time and that we can see the direction of

the current in each wire at this moment. The current direction is shown with arrows for each wire in Fig. 6, see i_1, …, i_6. The magnet lines of each current – ML_{i_1} , … , ML_{i_6} – are also shown for the three coils that are differently placed over the chip die. Two coils are vertical oriented, i.e. the inner surface of the first coil is in the *xz*-plane and the one of the second is in the *yz*-plane. The third coil is horizontally oriented, i.e. its inner surface is in the *xy*-plane. One of the vertical coils can feel the magnetic field only from green wires, the other vertical coil only from red wires. The horizontal coil can "feel" the magnetic field from green as well as from red wires. None of the shown coils can feel the magnetic field of the current through vertical wires, i.e. vias.

Fig. 7 shows schematically a coil placed over a die, i.e. over many thin and straight wires with alternating current. The coil and wires on the chip die are shown schematically but the relation of their sizes is realistic: the coil diameter is 150 µm; the wire thickness is shown for the IHP 250 nm technology.

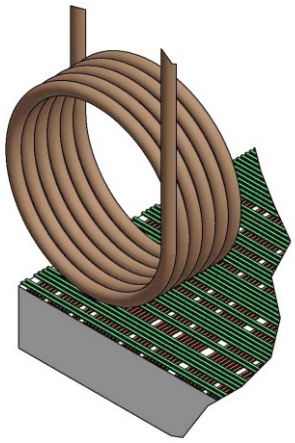


Fig. 7. A coil placed over a chip die: the coil and the chip die are shown schematically but the relation of their sizes is realistic.

The placement and the orientation of the coil in Fig. 7 are the best suitable ones for measurements of the magnetic field of "green wires" in the top metal layer and are not suitable for measurements of the magnetic field of "red wires".
Wires placed in deeper/nether layers, that have the same orientation as the one shown for "green wires", are characterized by an increased distance to the coil and their magnetic field is partially shielded by the "red wires".

The following practical aspects are consequences derived from the given above facts:

- the number of the coil turns determines also the height of the coil. If a coil has many turns *n*, it increases its sensibility according to (8), but each turn has its own distance to the wire (due to the diameter of the wire used in the coil) that causes a different magnetic flux through each coil turn
- with the goal to localize the measurements of the magnetic field, i.e. to reduce the impact of noise that is caused by surrounding electromagnetic radiation, the coil has to be shielded
- for measurements of the magnetic field over the chip die the distance from the coil to the chip can vary from almost 0 mm in the case of a decapsulated chip up to 0.5 mm that is the thickness of the package. When measuring from the backside, the distance from the coil to the wires is about 0.2-0.3 mm that is the thickness of the substrate of a decapsulated chip and about 0.5 mm in addition for a chip in a package.

3.1.3.2 EM probes available at IHP

The choice of the probes was done among those that are available at IHP and have suitable parameters. Their short description and parameters are given below.

Riscure EM Probes

The Riscure "high sensitivity" and "low sensitivity" EM probes were supplied to IHP with the EM Probe station in 2013. There was not much information provided regarding their technical parameters except the following. Both probes are active and contain an integrated amplifier with a bandwidth of 1 GHz. The amplifier has 3 amplification stages and provides a sensitivity of 100mV/1µT@1MHz for the "high sensitivity" probe or 20mV/ 1µT@1MHz for the "low sensitivity" probe.

Fig. 8. Riscure EM probes: "high sensitivity" Risc-EMPHS208 (top) and "low sensitivity" Risc-EMPLS211 (bottom).

The coil in the "high sensitivity" as well as in the "low sensitivity" probe has an inner area of 1 mm^2 and an outer area of 2 mm^2. There are no datasheets or other information available.

Recently, Riscure started to equip its EM measurement setup with near-field probes manufactured by Langer EMV-Technik GmbH [91], see for example the Riscure EM measurement equipment [92]. A short overview of some of Langer probes is given in the next section.

Langer EM *Probes*

The Langer EMV-Technik company was founded in 1998 and is located in Bannewitz, a suburb of Dresden city. It offers a variety of products for Electromagnetic interference analysis required during the development of printed circuit boards and integrated circuits.

The near-field EM probes produced by Langer are worldwide known and cover the frequency range up to 20 GHz. Such probes are utilized in most of the academic researches related to the measurements of electromagnetic radiation in the near-field region. Most of the probes are "small and handy" due to the pen shape and have an SMB jack output connector. In my experiments we used H-field probes only.

The EM probe **MFA-R 0.2-75** [93], shown in Fig. 9 is suitable for measuring magnetic fields in the range from 1 MHz up to 1 GHz.

Fig. 9. Langer MFA-R 0.2-75 EM probe.

The probe is active and therefore requires an external DC power to be provided through the bias tee for the integrated amplifier. It has a small probe head with a coil marked by a black dot and ensures a resolution of 300 µm. The coil sits in the plane of the shaft. The probe

can be used either as a vertical or horizontal one. Additional features include a current attenuating shell and therefore electrical shielding.

The EM probe **LFB-3** [94] near-field probe is shown in Fig. 10. The operating frequency range is 100 kHz to 50 MHz. The probe head has a diameter of 4 mm. It has shielding against electric field coupling and contains a coil placed orthogonally to the probe's shaft plane. The declared resolution is 2 mm.

Fig. 10. Langer LFB-3 EM probe.

The LF-B 3 is a passive probe and does not require any external power supply. In case when the amplification of measuring signals is required the probe may be connected to the input of an amplifier.

The near-field microprobes **ICR HH150-27** and **ICR HV 150-27** (see Fig. 11) are extremely sensitive and designed for high-resolution magnetic field measurements. Both have similar parameters. The operating frequency is in the range of 1.5 MHz up to 6 GHz. The probe head has a shielding against electric fields. The measuring coil has an internal diameter of 150 µm. It is located either horizontally within the ICR HH150-27 [95] probe head or vertically in the case of the ICR HV150-27 [96] probe. The ICR HH150-27 and HV150-27 probes have a resolution of 100 µm and 80 µm respectively.

a) *b)*

Fig. 11. Langer ICR HV150-27 (a), ICR HH 150-27 (b) EM probes and zoomed-in photos of the probe tips.

Probes have a bias tee powered amplifier integrated into the probe housing. Due to the fact that the probe tip is fragile and the optimal distance to the object being measured should be less than 1 mm, the manufacturer recommends the usage of probes in automatic positioning systems that have a collision protection function.

3.1.3.3 Selection of the best suitable probe, its placement and orientation

The selection of the best suitable probe among the probes described above was performed based on the attack results performed against the basic IHP ***kP*** design analysing EM traces captured during the execution of a ***kP*** operation on a Xilinx Spartan-6 FPGA. The input data, i.e. the value of the scalar k and EC point $\boldsymbol{P}$ were the same in all measurements. A measurement setup that was applied during this experiment as well as for the rest of the experiments conducted in this work is schematically depicted in Fig. 12.

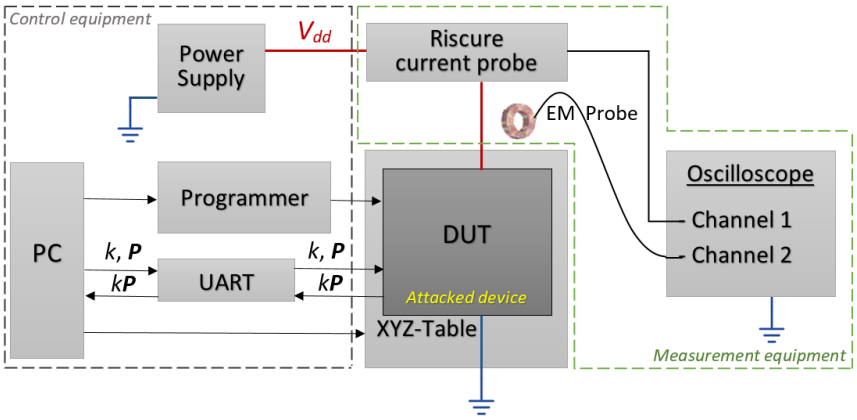


Fig. 12. A measurement setup for the SCA attacks in my experiments.

Each of the probes was placed close to the power supply line of the FPGA. As the signal/noise ratio can be used for selecting the best measurement position, we tried to find the highest signal range and the best signal-to-noise ratio for each of the probes while keeping the rest of the parameters such as design, inputs, supply voltage constant.

The attack performed against the measured traces is described in section 3.2.3. TABLE IV summarizes the attack results. The numbers in the table columns show the success of the attack. The attacks performed using the MFA-R 0.2-75 probe have the highest success: 14 extracted key candidates have the correctness higher than 90%. That means that at least 208 bits of the 232-bit long scalar k were extracted correctly. More details are given in [97]-[98].

TABLE IV. SUMMARIZATION OF THE ATTACK RESULTS USING DIFFERENT EM PROBES

Correctness	Number of extracted key candidates					
	Riscure HS	Riscure LS	MFA-R 0.2-75	LFB-3	ICR HV150-27	ICR HH150-27
>90%	-	-	14	5	-	-
80% to 90%	-	-	4	13	-	-
70% to 80%	7	15	2	6	-	-

Thus, the MFA-R 0.2-75 was chosen as the best suitable probe for EM trace measurements. Additionally, we experimented with different placements of the probe in order to find the best suitable place [27]. As a result, we selected a position close to the power decoupling capacitors of the device as the most suitable position. Measurements of electromagnetic emanations of ASICs were conducted over the core voltage wire.

3.2 Implementation of horizontal attacks

The statistical analysis for revealing a key exploits the basic statistical assumption: if a set consists of two subsets that can be distinguished from each other - the average values of both subsets are significantly different. Applied to a measured $k\boldsymbol{P}$ trace this means: if subsets of slots for processing the key bit value '0' can be distinguished from subsets for '1', the mean slot for '0'and the mean slot for '1' differ (significantly).

Thus, statistical tests can be easily applied by designers (white box cryptography), due to their knowledge of the processed scalar. The difference of means test as well as statistics using Pearson coefficients and method of least squares require a calculation of the mean slot for '0'and the mean slot for '1', i.e. they are not useful without the knowledge of the scalar k processed during the analysed $k\boldsymbol{P}$ trace.

The comparison to the mean test is a kind of statistical analysis adopted for the attackers as well. The assumption about the distinguishability of the '0'- and '1'-slots can be expressed as follows: the averaged (mean) profile of the processing of all key bits can be used as a distinguisher between the processing of key bit value '1' or '0'. Thus, the comparison to the mean allows to reveal an unknown processed scalar, i.e. it can be used by either attackers or by designers in case of evaluation of the design resistance.

In this section the statistical analysis methods implemented in this work that form the IHP SCA Toolkit are described and explained on examples. The difference of means test, the method of least squares and the correlation analysis using Pearson correlation coefficients are programmed in Excel. Other methods described in this section are programmed in Java. The method for the evaluation of the attack success is discussed in section 3.2.4. Section 3.2.6 shows an example of the analysis using the IHP SCA Toolkit.

3.2.1 **Representation of traces and evaluation of attack success**

In this thesis we use the following notations describing an analysis of ***kP*** traces attacking implementations using the Montgomery ladder, i.e. attacking ***kP*** designs for ECs over extended binary fields $GF(2^n)$:

- l – is the length (in bits) of the processed scalar $k = k_{l-1}k_{l-2}k_{l-3}\cdots \sum_{j=l-1}^{down\ to\ 0} k_j \cdot 2^j$; the maximum allowed length of scalars[12] used in ***kP*** operation for an EC over $GF(2^n)$ is n;
- j – is an index of a bit of the scalar k;
- in the main loop of the IHP ***kP*** designs the bits k_j with indexes $l\text{-}3 \geq j \geq 0$ are processed, i.e. (l-2) bits of the scalar k (see Algorithm 7 in section 2.4.1);
- each time interval in a measured or simulated trace of a ***kP*** execution corresponding to the processing of a single bit of the scalar k is denoted as a *slot*;
- CC – is the number of samples in each clock cycle; this number depends on the clock cycle frequency and on the sampling rate selected for measurements/simulations;
- cc – is an index of a sample value within a clock cycle;
- each slot is $\#ClocksInSlot$=54 clock cycles long;
- each slot contains $S=\#ClocksInSlot \cdot CC$ samples;
- each slot starts with a calculation of the 1st partial product of the 1st field multiplication in the corresponding main loop and finishes calculating the last partial product of the last field multiplication in the main loop iteration;
- i – is an index of a sample within a single slot ($1 \leq i \leq S$);
- L – is the number of '0'-bits processed in the main loop;
- $N=l\text{-}2\text{-}L$ – is the number of '1'-bits processed in the main loop.

According to these notations:

- $\boldsymbol{v}_j$ – is the slot with the index number j:
  - it corresponds to the processing of the j^{th} bit of the scalar k (i.e. to the processing of a k_j);

[12] Not randomized scalars are meant here. The length of randomized scalars can be higher than n.

- it consists of S=54·CC sample values: they are numbered from 1 to S: $1 \le i \le S$, i.e. $\boldsymbol{v}_j$ is an S-dimensional vector[13], or a one-dimensional array containing S elements;

- v_j^i – is the value of an i^{th} sample in the slot $\boldsymbol{v}_j$;
- *start sample* – is the sample v_{l-3}^1, i.e. it is the first sample in the first slot of the main loop[14] ;

Fig. 13 shows the part of the measured power trace that corresponds to the processing of the key bits k_{229} =0, k_{228}=1 and k_{227}=0 and a clock cycle of the trace, zoomed in. The scalar processed during the capturing of the trace is *k1* (see section 5.1). It is 232-bit long (*l=232*). In the main loop 230 bits are processed: 109 '0'-bits (*L*=109) and 131 '1'-bits (*N=131*). Each clock cycle consists of 625 sample values (*CC*=625). Each slot consists of 54·625=33750 sample values (*S*=33750). The processing of 230 bits in the main loop consists of: 230 slots, or 230·54=12420 clock cycles, or 230·54·33750=419175000 sample values.

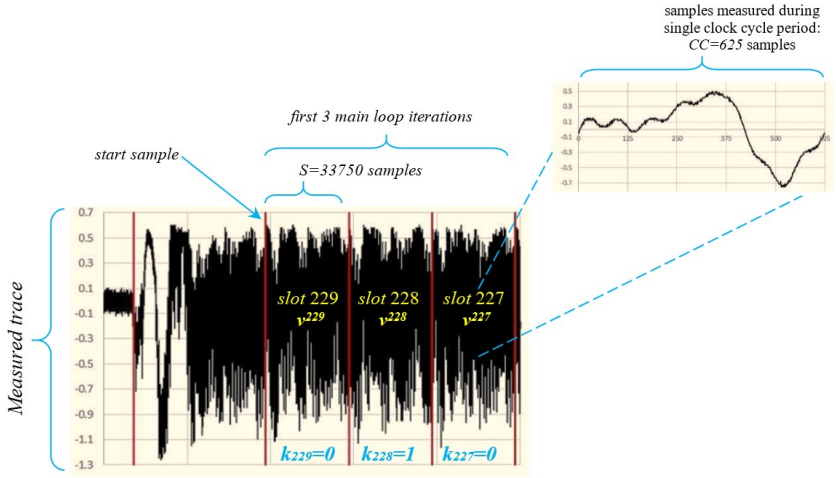


Fig. 13. Part of the measured power trace that corresponds to the processing of 3 key bits in the main loop. Power profile of the trace captured during a single clock cycle period is shown, zoomed in.

All samples with a fixed index number *i* in all (*l-2*) slots (i.e. all sample values v_j^i, with a fixed index *i* and index *j* in the range *l-3* ≥ *j* ≥ *0*) correspond to the same process (due to the regularity of the implemented Montgomery *k**P*** algorithm), i.e. the differences of these sample values can be used as a distinguisher of '0'- and '1'-shapes, i.e. for the extraction of a key candidate $k^{candidate\_i}$; thus we can obtain up to S key candidates $k^{candidate\_i}$ using statistical analysis.

Using statistical methods we extract the bit values of the scalar k which are processed in the main loop of the algorithm, i.e. we extracted all j^{th} bits, for *l-3* ≥ *j* ≥ *0* using all v_j^i sample values (with a fixed number *i*):

$$k^{candidate\_i} = k_{l-3}^{candidate\_i} k_{l-4}^{candidate\_i} \cdots \quad {}^{\_i} k_0^{candidate\_i} = \sum_{j=l-3}^{down\ to\ 0} k_j^{candidate\_i} \cdot 2^j \qquad (9)$$

Here $k_j^{candidate\_i}$ denotes the j^{th} bit of $k^{candidate\_i}$.

[13] A variable that denotes a vector or an array we mark using a ***bold italic font*** and variables that denotes a number/value we mark using *italic font (not bold)*

[14] This slot corresponds to the processing of k_{l-3} .

We evaluated the correctness of the extracted key candidates using the knowledge about the scalar *k* processed during capturing the analysed *k**P*** trace, i.e. we compared each $k^{candidate\_i}$ with the scalar *k*. The result of the comparison is the *relative correctness* of the $k^{candidate\_i}$ that we denoted as $\delta_1^{\,i}$. This is the number of the correctly revealed bits in the key candidate divided by all revealed bits, i.e.:

$$\delta_1^{\,i} = \frac{NumberOfCorrectRevealedBits(k^{candidate\_i})}{l-2} \cdot 100\% \tag{10}$$

For example if 200 bits of a key candidate were revealed correctly and the scalar *k* processed was 232-bit long, the correctness $\delta_1 = \frac{200}{232-2} \cdot 100\% \approx 87\%$.

The relative correctness $\delta_1^{\,i}$ is a value between 0% and 100%. If a key candidate $k^{candidate\_i}$ has the correctness $\delta_1^{\,i} = 100\%$ it means that the key candidate is equal to the processed scalar *k*. Correctness equal to 0 percent means that all bits of the revealed key candidate are wrong. This means also that our assumption used for the extraction of the key candidate is wrong and the opposite assumption needs to be correct. In this case we can easily obtain the key performing a bitwise inversion of the key candidate. Taking this fact into account we can calculate the relative correctness as a value between 50 and 100% as follows:

$$\delta^i = 50\% + \left|50\% - \delta_1^{\,i}\right| \tag{11}$$

Please note that an attack resulting in key candidates with a correctness of 50 percent is the worst-case from an attacker's point of view. This means that the statistical method used for the analysis cannot even provide a slight hint whether the key bit processed is more likely a '1' or a '0', i.e. this means that the attack was not successful at all. The worst-case from the attacker's point of view is the ideal case from the designer's point of view. We denote it as the *ideal case* in the rest of this thesis.

To simplify our analysis we represented each clock cycle using only one value – i.e. we compressed the simulated as well as measured traces. We experimented with different compression methods [99], i.e. for a clock cycle representation we took:

1 –the sample with the maximum value within the clock cycle period;
2 –the sample with the minimum value within the clock cycle period;
3 –the difference of the maximum and minimum values within the clock cycle period;
4 – the sum of all sample values simulated/captured during the clock cycle period;
5 – the sum of absolute values for all samples within the clock cycle period;
5 – the average of all sample values (arithmetic mean) within the clock cycle period;
6 – the sum of squared values of all samples within the clock cycle period.

After the compression, each clock cycle is represented as a single value and each slot contains *S=#ClocksInSlot=54* values only.

All these compression methods were implemented in Java and are a part of the IHP SCA Tool. We obtained the best attack success results (analysing the same *k**P*** trace) when representing each clock cycle as:

- the sum of sample values within the clock cycle (mostly applied for simulated traces);
- the sum of squared values of all samples within the clock cycle (mostly applied for measured traces).

In the next sections we describe simple statistical methods implemented for the analysis (see sections 3.2.2 and 3.2.3) and discuss their results in section 3.2.4.

3.2.2 **Analysis requiring knowledge of the processed scalar**

3.2.2.1 Difference of means

The main idea of the horizontal attacks is to distinguish parts of the trace corresponding to the processing of key bits '1' from parts corresponding to the processing of key bits '0' using for example statistical analysis methods.

Designers can use their knowledge about the processed scalar k to split the analysed trace into two sets of slots, one representing the processing of '1' and the second one representing the processing of '0'. The distinguishability of the '0'-slots from the '1'-slots is the measure allowing to evaluate the resistance of the implementation against selected horizontal attacks such as simple SCA or differential address-bit DPA. In an ideal case, from the designer's point of view, these two sets are not distinguishable, i.e. the mean shape of all '0'-slots and mean shape of all '1'-slots are, for example, equal or the shapes of all slots are fully random. Designers can apply for example Welch's test [100] to a big set of $k\boldsymbol{P}$ traces (as well as to a single $k\boldsymbol{P}$ trace) measured with different (but known) scalars k with the goal to evaluate statistically the distinguishability of the '0'-slots and the '1'-slots.

The difference of the means test that we implemented is a kind of the Welch's test applied to a single trace of a $k\boldsymbol{P}$ execution. Corresponding to our notation in section 3.2.1 the part of the scalar k processed in the main loop of the Montgomery ladder contains L key bits '0' and N key bits '1'.

We selected only slots corresponding to the processing of a key bit value '0' from the analysed trace and numbered them anew, from 1 to L. We calculated their average profile, i.e. the mean slot $\boldsymbol{mean_0}$:

$$\boldsymbol{mean_0} = \left\{ m_0^{\,1}, \ldots \right\}, \text{ with } m_0^{\,i} = \frac{1}{L} \sum_{n=1}^{L} v_n^i \tag{12}$$

Using the slots corresponding to the processing of a key bit value '1' we calculated their average profile, i.e. mean slot $\boldsymbol{mean_1}$:

$$\boldsymbol{mean_1} = \left\{ m_1^{\,1}, \ldots \right\}, \text{ with } m_1^{\,i} = \frac{1}{N} \sum_{n=1}^{N} v_n^i \tag{13}$$

After the calculation of average profiles for processing a key bit value '0' and a key bit value '1' we calculated their confidential intervals, i.e. we calculated the empirical standard deviations $\boldsymbol{s_0}$ and $\boldsymbol{s_1}$. The corresponding formulae can be found in [18], section 5.4 or in [101], section 4.6. Please note that we try here to explain the difference of means (DoM) method on the example of our analysed traces. A detailed description of DoM methods can be found in [101] or in [18]. In [18] the DoM test was applied for preparation to localized EMA attacks against a $k\boldsymbol{P}$ implementation for a Spartan-3a FPGA.

Using the calculated mean slots $\boldsymbol{mean_0}$ and $\boldsymbol{mean_1}$ and their confidential intervals the difference of both mean slots – the difference of means $\boldsymbol{(mean_0 - mean_1)}$ – can be calculated, with its confidential interval $\pm \boldsymbol{s_{0\text{-}1}}$.

Fig. 14 shows an example of the calculated mean slots $\boldsymbol{mean_0}$ (see solid green line) and $\boldsymbol{mean_1}$ (see solid red line) with their confidential intervals (dotted lines) calculated for a compressed measured power trace of the IHP basic $k\boldsymbol{P}$ design. The processed scalar k was 232-bit long. In the main loop of the Montgomery ladder109 '0'-bits and 121 '1'-bits were processed, i.e. S=54, L=109, N=121.

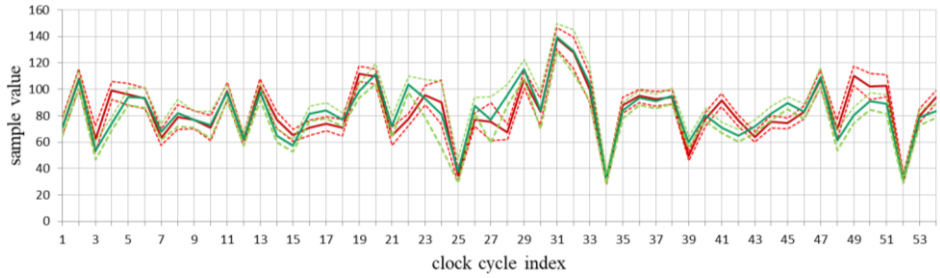


Fig. 14. An example of the calculated mean slots $\boldsymbol{mean_0}$ (see solid green line) and $\boldsymbol{mean_1}$ (see solid red line) with their confidential intervals (dotted lines).

The calculated difference of means is represented as a solid black line in Fig. 15; the confidential interval $\pm\boldsymbol{s_{0\text{-}1}}$ is represented with grey lines, symmetrically to the *x*-axis.

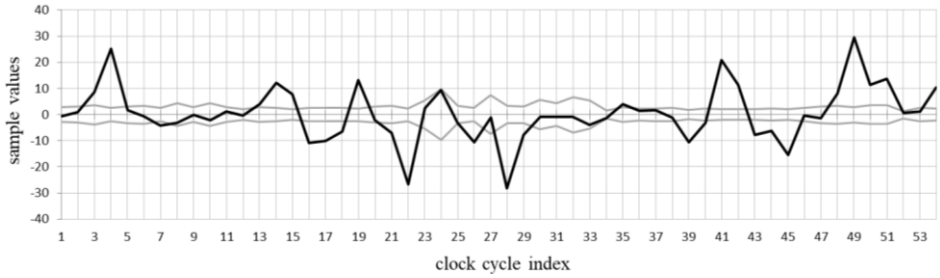


Fig. 15. Difference of means $\boldsymbol{(mean_0 - mean_1)}$ (see black line) with its confidential interval (see green lines) calculated using the measured power trace of the analysed $k\boldsymbol{P}$ execution.

If the processing of key bit values '0' is not distinguishable from the processing of '1', their mean slots are identical and the difference of both means is 0. Thus, the ideal case from the designer's point of view is when the difference of means, represented in Fig. 15 with a black line, will match the *x*-axis. Using this assumption the confidential interval $\pm\boldsymbol{s_{0\text{-}1}}$ is represented with two lines symmetrical to the *x*-axis. If the investigated $k\boldsymbol{P}$ design has an SCA leakage source within a clock cycle, the difference of means differs significantly from the 0 in this clock cycle, i.e. it shows a value out of the confidential interval.

Fig. 15 shows that our design has many processes that are SCA source leakages, in 28 of 54 clock cycles, i.e.: 3, 4, 7, 13-19, 21, 22, 26, 28, 29, 35, 39-45, 48-51 and 54.

An important fact is that the difference of means test does not allow to reveal the processed key. It allows only to define the clock cycles with high SCA leakage sources. More applicable for a key extraction are the mean slots $\boldsymbol{mean_0}$ and $\boldsymbol{mean_1}$ with their confidential intervals.

All these calculations as well as the graphical representation of the calculated vectors were programmed in Excel.

3.2.2.2 Analysis using Pearson correlation coefficients

For two arrays $\boldsymbol{x}$ and $\boldsymbol{y}$ of the same length a coefficient of their linear relationship can be calculated. The Pearson correlation coefficient r_{xy} is often used in SCA attacks. r_{xy} can be calculated as follows [102]:

$$r_{xy} = \frac{\sum_{n=1}^{s}\left(x_n - \bar{x}\right)\left(y_n - \bar{y}\right)}{\sqrt{\sum_{n=1}^{s}\left(x_n - \bar{x}\right)^2 \cdot \sum_{n=1}^{s}\left(y_n - \bar{y}\right)^2}} \tag{14}$$

Here n is the current number of the elements in the investigated array and the values $\bar{x}$ and $\bar{y}$ are arithmetical means of the arrays $\boldsymbol{x}$ and $\boldsymbol{y}$.

In our analysis we implemented the Pearson coefficients calculation for each slot in a compressed as well as in an uncompressed trace. For a compressed trace, if each clock cycle is represented using only a single value, that results in $s=S=54$. For an uncompressed trace s is equal to the number of samples in a single slot S of the analysed $k\boldsymbol{P}$ trace.

In our implementation we used the knowledge about the processed scalar k, i.e.[15]:

- We separated all slots in each attacked trace into two subsets: L slots corresponding to the processing of a key bit value '0' and $N=l-2-L$ subsets corresponding to the processing of a key bit value '1' of the processed scalar k. For each subset we calculated its mean slot, the $\boldsymbol{mean_0}$ and the $\boldsymbol{mean_1}$ as described in section 3.2.2.1.
- We obtained the key candidate as follows: for each slot $\boldsymbol{v_j}$ (with $l-3 \geq j \geq 0$) of the analysed $k\boldsymbol{P}$ trace we calculated two Pearson coefficients:
  - the $r_{\boldsymbol{v_j}\ \boldsymbol{mean_0}}$ using the $\boldsymbol{v_j}$ as the $\boldsymbol{x}$ array and $\boldsymbol{mean_0}$ as the $\boldsymbol{y}$ array,
  - the $r_{\boldsymbol{v_j}\ \boldsymbol{mean_1}}$ using the $\boldsymbol{v_j}$ as the $\boldsymbol{x}$ array and $\boldsymbol{mean_1}$ as the $\boldsymbol{y}$ array.
- We assess the value of the j^{th} bit of the key candidate by comparing the two calculated Pearson coefficients: if $r_{\boldsymbol{v_j}\ \boldsymbol{mean_1}} > r_{\boldsymbol{v_j}\ \boldsymbol{mean_0}}$ we assume that $k_j^{candidate}=1$; otherwise $k_j^{candidate}=0$.

The application of the Pearson correlation coefficients described here allows to obtain only one key candidate when attacking a $k\boldsymbol{P}$ trace.

The number of the key candidates can be increased by attacking an uncompressed trace if each slot will be regarded as a set of arrays of clock cycles samples. For example, in our implementation for the EC *B-233* the slot is always 54 clock cycles long, whereby each clock cycle can contain CC samples (for a compressed trace $CC=1$, for an uncompressed trace $CC>1$). We observed each slot $\boldsymbol{v_j}$ of an uncompressed $k\boldsymbol{P}$ trace as a set of 54 arrays $\left(\boldsymbol{v_j}\right)^i$, whereby each of the $\left(\boldsymbol{v_j}\right)^i$ contains CC elements:

$$\boldsymbol{v_j} = \left\{\left(\boldsymbol{v_j}\right)^1, \left(\boldsymbol{v_j}\right)^2, \ \ldots \ \left(\boldsymbol{v_j}\right)^i \ \ldots \ \left(\boldsymbol{v_j}\right)^{54}\right\} \tag{15}$$

Similar, we represented the mean slots $\boldsymbol{mean_0}$ and $\boldsymbol{mean_1}$ as a set of 54 arrays too:

$$\boldsymbol{mean_0} = \left\{\left(\boldsymbol{mean_0}\right)^1, \left(\boldsymbol{mean_0}\right)^2, \ \ldots \ \left(\boldsymbol{mean_0}\right)^{54}\right\} \tag{16}$$

$$\boldsymbol{mean_1} = \left\{\left(\boldsymbol{mean_1}\right)^1, \left(\boldsymbol{mean_1}\right)^2, \ \ldots \ \left(\boldsymbol{mean_1}\right)^{54}\right\} \tag{17}$$

Thus, each array $\boldsymbol{(mean_0)^i}$ ($1 \leq i \leq 54$) represents a clock cycle in the mean slot $\boldsymbol{mean_0}$. Similar, each $\boldsymbol{(mean_1)^i}$ represents a clock cycle in the mean slot $\boldsymbol{mean_1}$.

To obtain a key candidate i we calculated for each clock cycle $\left(\boldsymbol{v_j}\right)^i$ that we used as $\boldsymbol{x}$ array two Pearson coefficients:

- the $r_{\left(\boldsymbol{v_j}\right)^i\ \boldsymbol{mean_1}^i}$ using the $\boldsymbol{(mean_1)^i}$ as the array $\boldsymbol{y}$
- the $r_{\left(\boldsymbol{v_j}\right)^i\ \boldsymbol{mean_0}^i}$ using the $\boldsymbol{(mean_0)^i}$ as the array $\boldsymbol{y}$.

[15] Alternatively, autocorrelation analysis can be used to for the analysis without knowledge of the processed scalar.

We obtained for each *i* a key candidate $k^{candidate\_i}$ using the following assumption: the j^{th} bit of the key candidate is 1 if $r_{(v_j)^i\ \boldsymbol{mean_1}^i} > r_{(v_j)^i\ \boldsymbol{mean_0}^i}$; else the j^{th} bit of the i^{th} key candidate is 0:

$$k_j^{candidate\_i} = \begin{cases} 1, & if \quad r_{(v_j)^i\ \boldsymbol{mean_1}^i} > r_{(v_j)^i\ \boldsymbol{mean_0}^i} \\ 0, & if \quad r_{(v_j)^i\ \boldsymbol{mean_1}^i} \le r_{(v_j)^i\ \boldsymbol{mean_0}^i} \end{cases} \tag{18}$$

The Pearson correlation coefficients used in this way allow us to obtain 54 key candidates attacking a single ***kP*** trace.

3.2.2.3 Applying method of least squares for the trace analysis

Similar to the analysis using the Pearson correlation coefficients described in the previous section, we calculated the linearity coefficients corresponding to the method of least squares:

$$r\_LS_{xy} = \frac{\sum_{n=1}^{s}\left(x_n - \bar{x}\right)\left(y_n - \bar{y}\right)}{\sum_{n=1}^{s}\left(x_n - \bar{x}\right)^2} \tag{19}$$

We used in (19):

- each slot $\boldsymbol{v_j}$ (with *l-3* ≥ *j* ≥ *0*) containing *s*=*S* samples as the ***y*** array and ***mean₀*** (see section 3.2.2.1) as the ***x*** array for the calculation of $r\_LS_{v_j\ \boldsymbol{mean_0}}$
- each slot $\boldsymbol{v_j}$ (with *l-3* ≥ *j* ≥ *0*) containing *s*=*S* samples as the ***y*** array and ***mean₁*** (see section 3.2.2.1) as the ***x*** array for the calculation of $r\_LS_{v_j\ \boldsymbol{mean_1}}$

We decided the value of a j^{th} bit of the key candidate by comparing the two calculated linearity coefficients, if $r\_LS_{v_j mean_1} > r\_LS_{v_j mean_0}$, we assume that $k_j^{candidate}$=1, otherwise $k_j^{candidate}$=0. Thus, we obtained a single key candidate $k^{candidate}$.
We obtained 54 key candidates using formulae (15), (16) and (17) for the representation of each slot $\boldsymbol{v_j}$ and mean slots ***mean₀*** and ***mean₁*** and applying criterion (18) for the decisions. The calculation process is described in detail in the previous section; the difference is only that we applied the linearity coefficients $r\_LS_{(v_j)^i\ \boldsymbol{mean_1}^i}$ and $r\_LS_{(v_j)^i\ \boldsymbol{mean_0}^i}$ calculated corresponding to (19) instead of the Pearson correlation coefficients $r_{(v_j)^i\ \boldsymbol{mean_0}^i}$ and $r_{(v_j)^i\ \boldsymbol{mean_1}^i}$ calculated corresponding to (14).

3.2.3 **Comparison to the mean**

Assuming that '0'-slots differ (even slightly) from '1'-slots, an attacker can exploit the null hypothesis to reveal the key. In the case the '0'- and '1'-populations are distinguishable their mean shapes are different, i.e. ***mean₀*** ≠ ***mean₁***. The attacker cannot calculate both mean slots because the processed scalar is not known to him and his goal is to reveal it. But even

without any knowledge about the key bit value processed in each of the slots the attacker can still calculate the mean shape "***mean***" of all slots:

$$\boldsymbol{mean} = \{m^1, \ldots \quad \}, \text{ with } m^i = \frac{1}{l-2}\sum_{n=0}^{l-3} v_n^i, \text{ for } \forall i \in \{1,2,\cdots \quad \} \tag{20}$$

The attacker can try to distinguish the '0'-slots from the '1'-slots based on the assumption that the ***mean*** shape of both populations is between both shapes:

$$\boldsymbol{mean_1} < \boldsymbol{mean} < \boldsymbol{mean_0} \tag{21}$$

or

$$\boldsymbol{mean_0} < \boldsymbol{mean} < \boldsymbol{mean_1} \tag{22}$$

Attacker compares sample-wise each current slot *j* with the mean slot and reveals *S* key candidates for example according to the assumption (21) as follows:

> The j^{th} bit of the key candidate $k^{candidate\_i}$ is '1' if in the slot with number *j* the sample value with a number *i* is smaller than the sample value with the same number *i* in the ***mean*** slot. Else the j^{th} bit of the i^{th} key candidate is equal to '0':
>
> $$k_j^{candidate\_i} = \begin{cases} 1, & if \quad v_j^i < m^i \\ 0, & if \quad v_j^i \geq m^i \end{cases} \tag{23}$$

The assumption (23) allows us to obtain *S* key candidates.

The assumption (22) is opposite to the assumption (21) and can be expressed as:

$$k_j^{candidate\_i} = \begin{cases} 0, & if \quad v_j^i < m^i \\ 1, & if \quad v_j^i \geq m^i \end{cases} \tag{24}$$

The assumption (24) allows us to obtain *S* additional key candidates. Thus, the attacker obtains *2S* key candidates. Please note that *S* key candidates obtained using the assumption (24) can be derived from the *S* key candidates, obtained using the assumption (23), by their bitwise inversion. This fact is already taken into account when evaluating the attack success calculating the correctness δ^i in the range of 50% to 100% corresponding to (11), see section 3.2.1. Due to this fact, only one assumption (23) or (24) can be used to extract all key candidates.

Evaluating the success of an attack using the correctness δ^i is by far simpler for the designers, as they know the processed scalar *k* and can compare each of the key candidates bitwise with the scalar *k* and use this to calculate the success rate and to determine exactly which bits of the scalar were determined correctly. The relative correctness δ^i shows the success of the performed attack as well as the clock cycles with strong SCA leakage. Especially this aspect is important for designers because it can be helpful for localizing process(es) causing the SCA leakage.

The attacker can conclude if one of the extracted key candidates is equal to the processed scalar *k* or not calculating the EC point multiplication $k\boldsymbol{G}$ (where $\boldsymbol{G}$ is the base point of the EC) and comparing the results with the public key of the attacked person/device. Important for the attacks using the *comparison to the mean* approach is the knowledge about implementation details such as the start, the end and the duration of each slot. An additional assumption is that all the slots have the same duration. In comparison to designers, attackers do not have any knowledge about the start and duration of slots, i.e. they even need to speculate about these simple but very important details. A wrong separation of the measured $k\boldsymbol{P}$ trace into slots can reduce the attack success significantly.

3.2.4 Selection of the most suitable analysis method

We experimented with all statistical methods described above with the goal to select the "most suitable" one. We analysed *k**P*** traces measured/simulated using different sampling rates. We analysed uncompressed as well as compressed traces and we applied different compression methods. In many of our experiments the attack success analysing a compressed *k**P*** trace[16] using the comparison to the mean method was the best.

Fig. 16 shows the success rate of horizontal attacks against the same measured *k**P*** trace using the *comparison to the mean* performing a "direct comparison" of the extracted key candidates (violet dotted line) and the comparison to the mean performing "full comparison" of the extracted key candidates (red line).

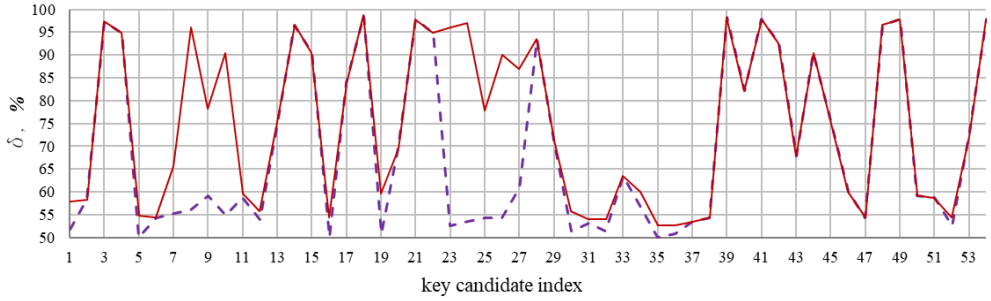


Fig. 16. Success rate of the horizontal attack based on statistical analysis using the comparison to the mean performing "direct comparison" of the extracted key candidates (violet dotted line) and the comparison to the mean performing "full comparison" of the extracted key candidates (red line).

The "direct comparison" and the "full comparison" differ in the counting of the $NumberOfCorrectRevealedBits(k^{candidate\_i})$ for the calculation of the relative correctness δ. In the "direct comparison" each key bit j of the extracted key candidate will be compared with the value of the j^{th} bit of the processed scalar k, i.e. it will be examined if $k_j^{candidate\_i} = k_j$.

Additionally, we can take the fact into account, that in the current slot not only the data related to the current key bit but also the previous or the next key bit can be processed. For example in parallel to the field multiplication processing the current key bit, a write to register operation that is related to the processing of the previous (or the next) key bit can be performed. Thus, we can increase the attack success by comparing the extracted key candidate with the scalar k shifted 1 bit to left (i.e. it will be examined if $k_j^{candidate\_i} = k_{j-1}$) or with the scalar k shifted 1 bit to right (i.e. it will be examined if $k_j^{candidate\_i} = k_{j+1}$). The number of key bits that can be compared directly is *l-2*. Comparing with the shifted scalar k is possible for *(l-3)* bits. For each of the 3 comparisons we obtained its $NumberOfCorrectRevealedBits(k^{candidate\_i})$ that results in 3 different correctnesses δ_{direct}, $\delta_{shiftedToLeft}$ and $\delta_{shiftedToRight}$. For each key candidate, we selected the best of these 3 correctnesses for the attack success evaluation:

$$\delta^i = \max\left(\delta^i{}_{direct}, \delta^i{}_{shiftedToLeft}, \delta^i{}_{shiftedToRight}\right). \quad (25)$$

We denote this evaluation approach as the "full comparison" of the extracted key candidates and used it in the rest of this thesis, due to the fact that its increases the attack success.

Comparing the attack results using different statistical methods and using our experience with the analysis of different traces we can conclude that the comparison to the mean is the most practical one. It does not mean that this method is the best in all cases. For example, it

[16] The most often applied compression method was sum of squared sample values.

is effective if the number of '0'-slots L is quite equal to the number of '1'-slots N. Otherwise, i.e. if L and N differ significantly, the *comparison to the mean* can be not so effective [103].

Additionally, we compared statistical analysis methods with selected machine learning methods [32]-[35], i.e. with a Principal Component Analysis and with a *k-means*. We did not observe an increasing attack success rate using the machine learning approaches in comparison to the usually applied statistical methods, especially for the designs with countermeasures. Thus, we experimented with other relatively simple approaches with the goal to evaluate the resistance of the designs against horizontal attacks. We automated simple SCA analysis as well as an attack based on fast Fourier transform. These additional analysis methods can be applied to the uncompressed traces and were most often used after the comparison to the mean approach. Their description is given in the next section.

3.2.5 **Improvements of the analysis**

3.2.5.1 Automated Simple SCA instead optical inspection of traces

A simple analysis attack was presented by Kocher [70] in 1999 on the example of DES. Later SPA attacks were shown for AES [104], RSA [36], and ECC [48]. The basic idea behind simple analysis attacks is the possibility to distinguish different sets of secret scalar-dependent instructions from power, timing or others side-channels. An attacker tries to reveal the secret scalar via simple optical inspection of a single measured trace during the execution of a cryptographic operation.

The main difference to the statistical analysis attacks such as for example the *comparison to the mean* (see section 3.2.3) is that here we do not calculate any statistical parameters such as mean values, variation, etc. The difference to usual simple analysis attacks is that we automated our attack with the goal to make the attack effective and fast. We implemented this approach in Java and applied it for the analysis of different IHP designs[28], [105]-[106].

Usually, when running simple power analysis or simple electromagnetic analysis an attacker assumes that the attacked design is a bitwise processing of the secret binary number, i.e. the *key*, whereby the processing of a key-bit value '0' differs from the processing of a key-bit value '1'. Due to these assumptions, the attacker can apply the following principles in simple analysis attacks:

- The shapes of the processing of key bit values '0' look similar to each other;
- The shapes of the processing of key bit values '1' look similar to each other;
- The shapes of the processing of key bit values '0' are distinguishable from the shapes of the processing of key bit values '1' using simple visual inspection;
- The analysed trace is a set of two different kinds of shapes, i.e. a sequence of '0'-shapes and '1'-shapes;
- The correctly recognized/extracted/revealed sequence of '0'- and '1'-shapes corresponds to the used secret, i.e. to the scalar k.

Thus, when executing a simple analysis attack the attacker looks at the measured trace (an optical inspection only) and tries to apply the above-listed principles, i.e. the instruments used for the analysis are the eyes of the attacker and his natural intelligence. Due to this fact, the success of simple analysis attacks depends directly on the distinguishability of the '0'- and '1'-shapes. If the difference is significant and can be easy seen the sequence of the shapes in the trace is clear. The key candidate extracted using this clear sequence matches with the real processed key 100%, i.e. all bits of the key are revealed correctly. The number of correctly revealed key bits can be used as a criterion of the success of the attack (see correctness δ in section 3.2.1) if the attacker knows the actual processed scalar k. The attacker knows the pro-

cessed scalar only if he analyses the trace measured during signature verification corresponding to ECDSA[17]. This is feasible as the scalars processed when verifying a signature are not secret, i.e. they can be derived from a part of the signature transmitted as a plain text. Designers can use their knowledge about the scalar processed for testing the resistance of their designs, i.e. we have the possibility to compare each extracted key candidate $k^{candidate}$ with the really processed key k to calculate the correctness δ of the key candidate.

It is obvious that the success of the simple analysis attack depends extremely on the length of the analysed trace, distinguishability of '0'- and '1'-shapes in the trace as well as on the experience of the attacker, whereby the more regular the implemented algorithm is - the less distinguishable are '0'- and '1'-shapes. Even in the case of white-box cryptography, i.e. if a designer with the full knowledge of the implementation details and the processed key tries to perform a simple analysis attack, the simple analysis attack can be a complex task. In the first step the attacker tries to separate the trace into slots, i.e. into parts that correspond to the processing of a single key bit. The separation of the trace into slots can be a complex and non-trivial task, especially for non-experts. If slots are successfully separated the attacker tries to classify them into "similar" and "different" slots, i.e. the attacker compares the slots with each other and then extracts the sequence of '0'- and '1'-shapes. Fig. 17 shows the zoomed-in traces separated into slots as well as the processed key bit values.

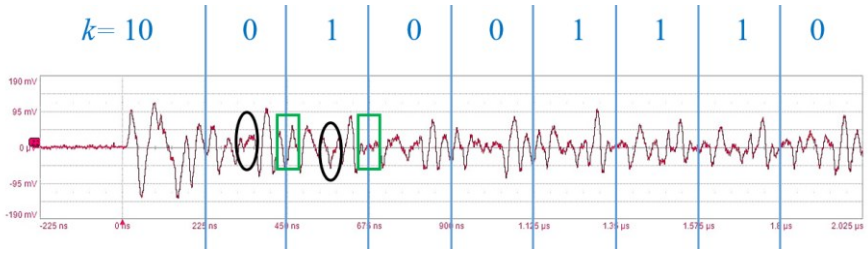


Fig. 17. SPA of an IHP ***kP*** design: power trace of a ***kP*** execution on an FPGA.

After splitting the traces into slots the '0'- and '1'-shapes in Fig. 17 can be easily distinguished from each other. The distinguishability "symptoms" are in the middle of the slots (see black circles in Fig. 17) and at the end of the slots (see green rectangles in Fig. 17).

An automation of the SPA allows to perform the analysis of the trace fast and certain: it allows to "magnify" the differences. Thus, we automated the analysis. We separated the analysed trace into slots and overlapped all shapes. The visualization of the analysis is given in Fig. 18. We marked all '0'-shapes blue and all '1'-shapes orange. Each slot contains S=9000 samples.

Fig. 18 illustrates how different '0'- and '1'-shapes are. The distinguishability "symptoms" can also be easily seen in the middle and at the end of the slots. Moreover, there are parts in the slots in which the set of all '0'- shapes is completely separated from the set of '1'-shapes. These parts are marked with black rectangles in Fig. 18 and are shown zoomed in, in the upper part of Fig. 18.

[17] This knowledge, together with the measured traces, can be helpful to prepare different template attacks as well as attacks using artificial intelligence approaches, for example supervised as well as unsupervised machine learning algorithms, as analysis tool.

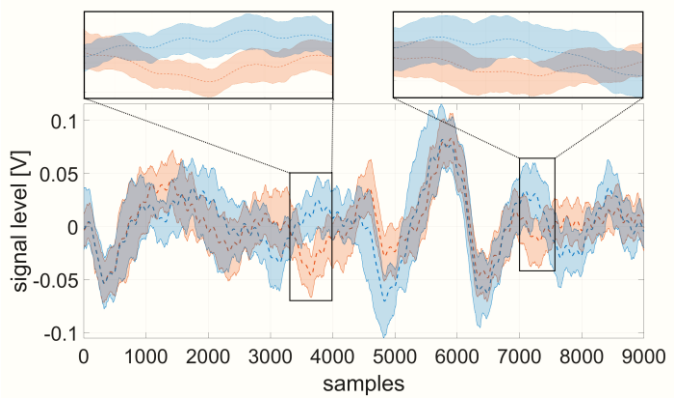


Fig. 18. Trace of the analysed ***kP*** design: the trace was partitioned into slots; all shapes were overlapped; all '0'-shapes are marked blue; all '1'-shapes are marked orange. The dark blue line represents the mean line for all blue lines; the red line represents the mean line for all orange lines.

We used the fact that the set of all '0'- shapes is completely separated from the set of '1'-shapes for automating the analysis, i.e. we programmed the recognition of such "gaps" between the sets of blue and orange lines. Each sample with such a "gap" allows to clearly distinguish all key bits '0' from all key bits '1' which causes the 100% success of the attack, i.e. the number of samples with "gaps" corresponds to the number of key candidates with a correctness δ=100%. Due to this fact we define the following additional evaluation criteria for a successful attack:

- number of samples with "gaps"
- maximal "gap" distance

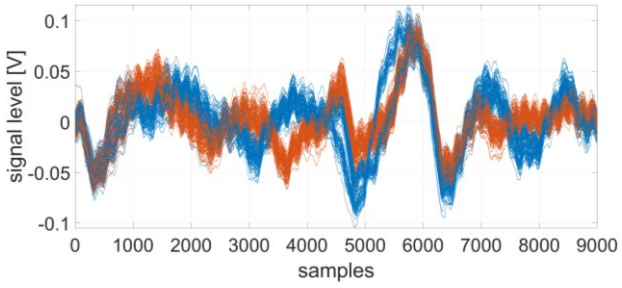
As designers we know the processed key, i.e. scalar k. We made a java application that splits the captured trace into slots and partitions them into '0'- and '1'-groups. We have L '0'-slots and N '1'-slots; each slot consists of S=9000 sample values. In this way we determine the upper and lower border for the set of blue lines and orange lines (see Fig.18), which represent '0'-shapes and '1'-shapes respectively

The next step was to check whether these two groups, i.e. '0'-slots and '1'-slots are overlapping or not, i.e. for $i \in \{1,2,\cdots$ the two groups can be distinguished from each other if:

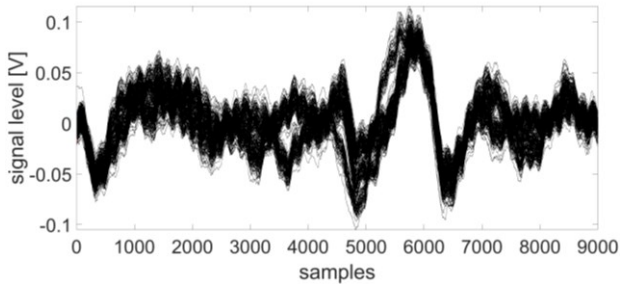
$$\max\_1^i < \min\_0^i \qquad \text{or} \qquad \max\_0^i < \min\_1^i \tag{26}$$

If one of the conditions (26) holds true, i.e. if the set of '0'-shapes was fully separated from the set of '1'-shapes, it means that the key can be successfully revealed. Depending on the design's implementation, finding points satisfying these conditions can be a non-trivial task. Especially for the designs where slots consist of millions of points, as it might be only one single point of such type within the slot. In this case detecting such points becomes infeasible by simple visual inspection of the trace. Our java application provides a list of slot point indexes where these conditions are met, i.e. it helps designers to find the leakage points and to identify leakage sources in their designs. Also attackers can exploit this fact for the key extraction.

This approach is visualized in Fig. 19.



a)



b)

Fig. 19. All slots of a measured $k\boldsymbol{P}$ trace in the same coordinates: *a)* – slots are separated into 2 groups, i.e. into '0'-slots (orange) and '1'-slots (blue); this separation is possible only using the knowledge about the processed scalar *k*. *(b)* all slots are shown without knowledge of the key, i.e. from the attacker's point of view.

Fig. 19-*a)* shows the slots partitioned into two groups according to the value of the key bit processed in each slot: the slots corresponding to the processing of a key bit '1' are marked in blue and the slots corresponding to the processing of a key bit '0' are marked in orange. This separation is possible only using the knowledge about the processed scalar.
All slots from the attacker point of view are shown in Fig. 19-*b,* i.e. all 230 slots are shown in the same coordinates, overlaying each other and all they are marked in black. Some "gaps" between sample values, for example for $i \in \{3500, \cdots$, can be successfully used for revealing the key.

3.2.5.2 FFT-based analysis

In this section we describe an analysis method based on fast Fourier transform (FFT), i.e. we try to distinguish the '1'-slots and '0'-slots no longer in the time domain but in the frequency domain.

According to the Nyquist criterion, the sampling frequency must be at least twice the maximum frequency component in the signal to prevent wrong signal representation due to aliasing. For example, if a trace is captured with a 2.5 GS/s sampling rate, all frequency components above 1.25 GHz have to be attenuated by using antialiasing low pass filters. The FFT-based methods require many samples within a clock cycle, i.e. they cannot be applied to a compressed trace. Such technical aspects can influence the success rate significantly and have to be taken into account.

We applied the FFT to each clock cycle of the analysed part of the trace, i.e. we calculated the spectrum of each clock cycle. After that, we represented each clock cycle as a sequence of $CC/2$ harmonics[18] with different spectrum amplitudes.

[18] The number of samples within the clock cycle is denoted as *CC*, see section 3.2.2.2

The analysed part of the trace, i.e. the *l-2* slots can be represented using values of harmonic's amplitudes: each j^{th} slot is an array of the harmonic's amplitudes:

$$(A_h)_j = \left\{ (A_h)_j^{\,1}, (A_h)_j^{\,2}, \cdots \; , (A_h)_j^{\,\#ClocksInSlot} \right\}, \tag{27}$$

where each $(A_h)_j^{\,i}$ for $i \in \{1,2,\cdots \quad ksInSlot\}$ is an array of harmonic's amplitudes representing the i^{th} clock cycle within the j^{th} slot, and $h \in \left\{1,2,\cdots \; \frac{CC}{2}\right\}$ is an index of the harmonics within a clock cycle.

Following the idea, that each single harmonic in one of *#ClocksInSlot*=54 clock cycles represents information about the key bit processed in the slot, we formed up to $\#ClocksInSlot \cdot \frac{CC}{2}$ sequences of harmonic's amplitudes as follows:

$$\begin{array}{l} for\ h{=}1\ to\ \frac{CC}{2} \\ \quad for\ i{=}1\ to\ \#ClocksInSlot \\ \quad\quad (A_h)_{l-3}^{\,i}, (A_h)_{l-4}^{\,i}, ..., (A_h)_0^{\,i} \\ \quad end\ for \\ end\ for \end{array} \tag{28}$$

We calculated the mean amplitude value $\overline{(A_h)^i}$ for each of the sequences:

$\overline{(A_h)^i} = \frac{1}{l-2}\left((A_h)_{l-3}^{\,i} + (A_h)_{l-4}^{\,i} + ... + (A_h)_0^{\,i} \right)$, and obtained the i^{th} key candidate using the following pseudocode:

$$\begin{array}{l} for\ h{=}1\ to\ \frac{CC}{2} \\ \quad for\ i{=}1\ to\ \#ClocksInSlot \\ \quad\quad for\ j{=}1\ to\ (l\text{-}2) \\ \\ \quad\quad\quad if\ (A_h)_j^{\,i} > \overline{(A_h)^i} \\ \\ \quad\quad\quad\quad k_j^{\,candidate\ h,i} = 1 \\ \quad\quad\quad else \\ \quad\quad\quad\quad k_j^{\,candidate\ h,i} = 0 \\ \quad\quad\quad end\ if \\ \\ \quad\quad end\ for \\ \quad end\ for \\ end\ for \end{array} \tag{29}$$

Thus, this method allows to extract $\frac{CC}{2} \cdot \#ClocksInSlot$ key candidates $k^{candidate\ h,i}$. To evaluate the success of the attack each extracted key candidate can be compared with the processed scalar *k* in exactly same way as for the comparison to the mean method, i.e. calculating a relative correctness δ for each key candidate.

The analysis method described here was implemented in Java and is a part of the IHP SCA Toolkit.

3.2.6 **Analysis of a trace using IHP SCA Tool**

We implemented all calculations described here – trace compression methods, statistical analysis methods, improving the attacks success evaluation, automated simple SCA as well

as FFT-based analysis including a Graphical User Interface (GUI) in Java as a part of the IHP SCA Tool. Only the DoM, Pearson correlation and least squares were implemented as Excel tables. The IHP SCA Toolkit simplifies and accelerates the attack process significantly. The toolkit allows to quickly evaluate the ***kP*** designs' resistance against a wide range of horizontal attacks and helps the designer to find the leakage sources in case of a successful attack.
The currently implemented functionality includes:

- Support of different sampling rates for the measured/simulated traces;
- Compatibility with different implementations of *regular* algorithms;
- Different trace compression techniques;
- Trace noise reduction options;
- Extraction of the *regions of interest* from the raw trace;
- Implementation of the attacks mentioned in sections 3.2.2, 3.2.3 and 3.2.5.

Fig. 20 shows parts of the GUI for the approaches developed by me.

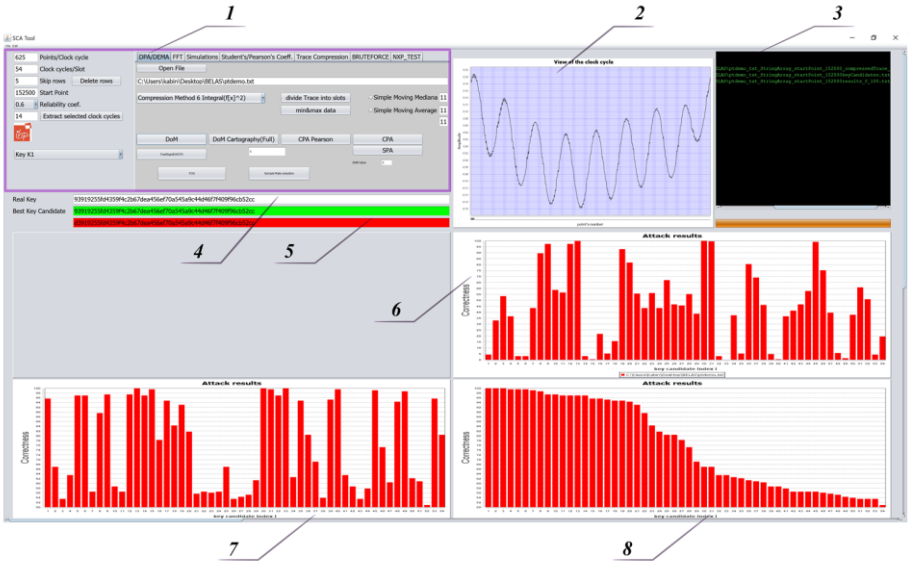


Fig. 20. Graphical User Interface for the IHP SCA Toolkit developed in this work.

The GUI region marked with 1 allows the user to specify necessary input parameters, required for the trace analysis, as well as to define trace preparation steps such as compression or noise reduction and the kind of the attack to be performed.
After starting the attack additional auxiliary information will be displayed:

- In region 2: the shape of the analysed trace for a single clock cycle.
- In region 3: the output console informs the user about any issues caused during the attack execution and provides information regarding paths where the analysis results are stored.
- The region marked with 4 contains the scalar k – the key – that was used during cryptographic operation whereas region 5 shows two key candidates revealed with the highest (close to 100%) and the lowest (close to 0%) correctness. In case of a full match to the applied key, the corresponding key candidate will be highlighted with green colour.
- Region 6 shows the key correctness of each of the key candidates calculated as a value between 0% and 100% (i.e. the correctness δ_1 is shown, see section 3.2.1).
- Region 7 provides the key correctness of each of the key candidates calculated as a value between 50% and 100%.

- Region 8 represents the correctness of all key candidates sorted in descending order.

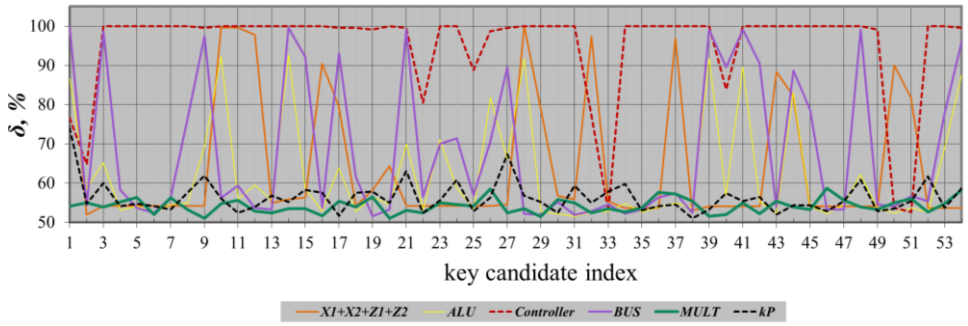
The information from regions 6 to 8 may help designers to identify the leakage source and eliminate it. The output data such as a compressed trace, key candidates and their correctness values as well as attack parameters are stored in a separate file.

The implemented IHP SCA Toolkit was demonstrated in 2017 and 2018 during the Technical Demonstrations Session of the International Conference on ReConFigurable Computing and FPGAs [107]-[108] and in 2020 at the VESS workshop of the Mediterranean Conference on Embedded Computing [109].

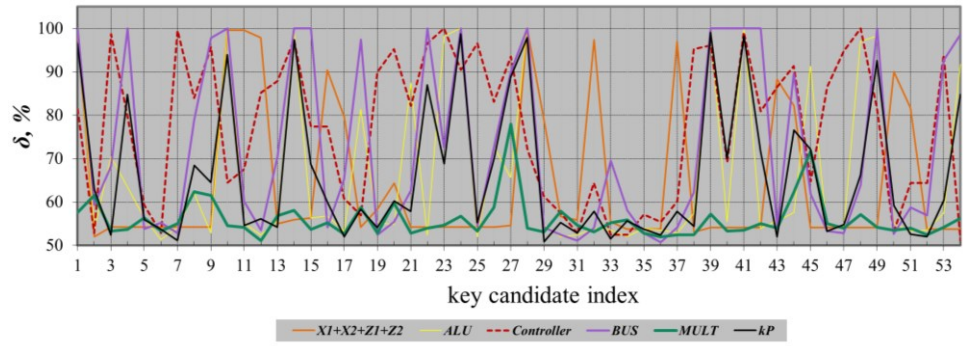
4 Attacks against the Basic design

In order to to verify and generalise some methods for implementing ***kP*** algorithms in hardware as proposed in [74], we synthesised the IHP basic ***kP*** design for the IHP 250 nm as well as for the IHP 130 nm. We used the same inputs as used in [74], and obtained power traces, which we attacked afterwards to evaluate the resistance of both design instances against a horizontal attack. We used the comparison to the mean method for the key extraction analysing the synthesised power traces, similar to [74]. The design synthesized for the IHP 250 nm was not resistant to the performed attack. The processed scalar k was revealed completely. Analysing the power trace for the design synthesized for the IHP 130 nm technology, the best extracted key candidate had a correctness of 71%, that is similar to results obtained in [74]. Thus, the success of the same attack differs significantly for the same VHDL code if synthesised for different technologies [110].

We analysed the 250 nm design in the same way as the 130 nm design with the goal to determine which component in the 250 nm design causes the high success of the performed attack and why the resistance of both designs differs so significantly. Fig. 21 shows the results of the analysis for the whole ***kP*** design and for its components: field multiplier ***MULT*** (see green line), registers X_1, X_2, Z_1, Z_2 (see orange line), ***ALU*** (see yellow line), ***Controller*** (see dotted red line) and ***BUS*** (see violet line).



a)



b)

Fig. 21. Horizontal attack using the comparison to the mean method: results of the analysis for the IHP basic ***kP*** design and its components: *a)* – synthesised for the IHP 130 nm technology; *b)* – synthesised for the IHP 250 nm technology.

Fig. 21 shows that:

– ***ALU***, registers, ***Controller*** and ***BUS*** are strong SCA leakage sources for designs synthesized in both IHP technologies;

– The block ***MULT*** is the most resistant block of the design (it is the most power and area-consuming component).
– The attack results using the PT of the whole ***kP*** design synthesised for the IHP 250 nm technology and those using the PT of ***BUS*** are very similar, compare black and violet lines.

Due to the fact that the addressing of blocks/registers in the Montgomery ***kP*** algorithm depends on the processed bit value k_j of the scalar k, horizontal (differential) SCA attacks can be successful. We evaluated the similarity of the traces to understand, why the IHP basic ***kP*** design synthesised for the IHP 130 nm technology is resistant and why the design synthesised for the IHP 250 nm technology is not resistant against the performed attack, as well as to determine the block(s) mostly influenced the shape of the power trace of the ***kP*** design. We calculated the correlation coefficients between the PTs of the whole ***kP*** design and its components for both technologies. TABLE V. shows the correlation coefficients.

TABLE V. PEARSON COEFFICIENTS CALCULATED FOR THE WHOLE *kP* DESIGN AND ITS COMPONENTS

kP (technology)	***MULT***	***ALU***	6 registers together: $X_1, X_2, X_3, X_4, Z_1, Z_2$	***Controller***	***BUS***	***rest***
kP (250 nm)	0.63	0.39	0.10	0.35	0.73	0.08
kP (130 nm)	0.75	0.44	0.16	0.18	0.76	-0.02

For both technologies, the ***BUS*** is the most influencing block, whereby it is an SCA leakage source. The Pearson coefficients are 0.73 and 0.76 for the 250 nm and 130 nm technology, respectively.

The next block that significantly influenced the shape of the whole ***kP*** design is the field multiplier ***MULT***. The Pearson coefficients are 0.63 and 0.75 for the 250 nm and 130 nm technology, respectively. It is the block with a high resistance against the performed attack.

The next block, influencing the shape of the power trace of the ***kP*** design is the block ***ALU*** (the Pearson coefficients are 0.39 and 0.44 for the 250 nm and 130 nm technology, respectively). Its resistance against performed attack is low.

Additionally, the block Controller, which is not resistant against the performed attack, influences the shape of the ***kP*** design synthesised for the IHP 250 nm technology significantly (the Pearson coefficient is 0.35). Its influence is comparable to the influence of the ***ALU***. For the design synthesised for the IHP 130 nm technology its influence is not so high, here its Pearson coefficient is 0.18.

Based on the results shown in Fig. 21 and TABLE V. we assumed that the activity of the ***BUS*** is the main SCA leakage source. If this assumption is true, the reason why the performed horizontal attack was successful, is the key-dependent addressing of the components for the *write-to-bus* and for the *read-from-bus* operations. The analysis of the operation flow in the main loop iteration of the implemented Montgomery ***kP*** algorithm confirms this assumption too (see Appendix 1).

Assuming, that the activity of the ***BUS*** is better hidden or compensated by the other components in the 130 nm technology than in the 250 nm design, we decided to concentrate on the following strategy to increase the inherent resistance of the ***kP*** design: reducing the relative contribution of the ***BUS*** to the PT of the whole design by increasing the field multiplier's contribution.

5 Evaluation of state-of-the-art countermeasures

In this section we investigate the influence of selected randomization countermeasures on the resistance of the $k\boldsymbol{P}$ design. We show that well-known randomization techniques such EC point blinding and key randomization are not effective against horizontal attacks. We evaluated also the resistance of a GALS-ified $k\boldsymbol{P}$ design against SPA. Opposite to [67] we were able to reveal the key, despite applying of 3 randomized clock signal frequencies. Additionally, we attacked an open-source $k\boldsymbol{P}$ design implementing unified formula for EC point doublings and point additions and randomization of the sequence of these operations in the Montgomery ladder. Attacking power traces simulated for the IHP 250 nm technology we were able to reveal the processed scalar completely.

5.1 Coron's Countermeasures

Three randomisation methods – the EC point blinding, the randomization of the projective coordinates of the EC point and the randomisation of the scalar k – were proposed in [48] as countermeasures against DPA (see section 2.3.2). They are well-known and effective countermeasures against vertical attacks, i.e. attacks using many (at least two) traces of $k\boldsymbol{P}$ executions. Unfortunately, the fact that these countermeasures are not effective against single-trace attacks was and not well-known. Such a misunderstanding can be dangerous because implementing even all Coron's countermeasures does not reduce the success of horizontal SCA attacks, for example of a simple power analysis using the comparison to the mean (see section 3.2). In [111] we demonstrated that the countermeasures based on the randomisation of input data are not effective against horizontal attacks.

For a fair evaluation of a single parameter/factor influencing the success of SCA attacks it is important that all other factors, that can influence the success rate remain constant. A possible way to evaluate all 3 randomization methods [48] using the same design and without any change of it, is implementing a design that supports all these 3 randomization possibilities. In this case it is possible to evaluate the influence of each randomization method as well as all their combinations using always the same design. A design supporting the countermeasures was not available in IHP.

The EC point blinding as well as the key randomisation proposed in [48] can be realized as a $k\boldsymbol{P}$ calculation with other – randomized – inputs, i.e. they do not require any changes of the design and can be evaluated using the basic IHP $k\boldsymbol{P}$ design. The randomisation of projective coordinates of the EC point $\boldsymbol{P}$ requires changes in the design that can influence the results of the SCA attacks significantly, i.e. in this case it is not possible to evaluate the influence of the "pure" randomisation method.

Due to the facts described above, we used the basic IHP $k\boldsymbol{P}$ design without any changes to demonstrate that the randomisation of the scalar k, blinding of the EC point $\boldsymbol{P}$ and their combination are not effective countermeasures against horizontal SCA attacks. We used the following randomly selected scalars and EC points in the experiments:

$k1$=93919255FD4359F4C2B67DEA456EF70A545A9C44D46F7F409F96CB52CC

$k2$=CDEA65F6DD7A75B8B5133A70D1F27A4D9506ECFB6A50EA526EB3D426ED

$\boldsymbol{R}$= (x_R, y_R) = (99FC5CE2CAFA210368FCCD13D8347B13648E5F6436F2BF8E12D2B2D0CC,
10B44430C0124A3009C67B13BD90BC379EAB04156658C64D5C0D0F9049F)

$\boldsymbol{S}$= (x_S, y_S) = (181856ADC1E7DF1378491FA736F2D02E8ACF1B9425EB2B061FF0E9E8246,
9FED47B796480499CBAA86D8EB39457C49D5BF345A0757E46E2582DE6)

Please note, that the scalar *k1* and the EC points ***S*** are the inputs mostly used in this work (see section 3.1.1).

We simulated power traces for the following *k**P*** executions:

- *k1*·***R***, i.e. a *k**P*** operation with a random scalar and a randomly selected EC point
- *k1*·***S***, i.e. a *k**P*** operation with the same scalar *k1* but with another EC point
- *k1*·(***R***+***S***), i.e. a *k**P*** operation with the same scalar *k1* and with EC point ***R*** that is blinded using EC point ***S***
- *k1*·(***R***+2***S***), i.e. a *k**P*** operation with the same scalar *k1* and with EC point ***R*** that is blinded using the doubled EC point ***S***

We simulated additional *k**P*** traces using the EC points described above but with the scalar *k2*:

- *k2*·***R***
- *k2*·(***R***+***S***)
- *k2*·(***R***+2***S***)

Additionally, we simulated power traces for *k**P*** executions with the randomized scalar *k1'*. It was calculated as the sum of the scalar *k1* and order *q* of the EC *B-233* basis point [19], i.e. for *k1'=k1+r·q*, with *r=1* (see explanation in footnote[20]):

- *k1'*·***R***
- *k1'*·***S***
- *k1'*·(***R***+***S***)

The algorithm investigated, i.e. the VHDL code, as well the target clock frequency is also the same in all experiments here. The simulations described above were performed for the IHP basic *k**P*** design synthesized for both technologies available at IHP, i.e. the IHP 250 nm and IHP 130 nm technologies [112]. Details about the simulation process are described in section 3.1.1. In the rest of this section the used technology is given as a part of the name of the simulated power trace. For example the power trace, which was simulated for experiment 1, is denoted as *k1·R_250nm*. In total 10 traces for each technology were simulated and statistically analysed, using the comparison to the mean method. The power traces were simulated for a clock cycle of 30 ns, i.e. for a clock frequency of about 33 MHz.

Fig. 22 and Fig. 23 show the attack results for all traces analysed in this experiment for the IHP 250 nm technology and for the IHP 130 nm technology, respectively. The white horizontal line at 50 percent shows the ideal case.

[19] In hexadecimal: q=1000000000000000000000000013E974E72F8A6922031D2603CFE0D7; in decimal q=6901746346790563787434755862277025555839812737345013555379383634485463, corresponding to [8].

[20] Only the value *r*=1 was possible to use for the key randomization experiments without any change of the attacked designs. Using *r*>1 requires increasing length for the register storing the value of the processed scalar.

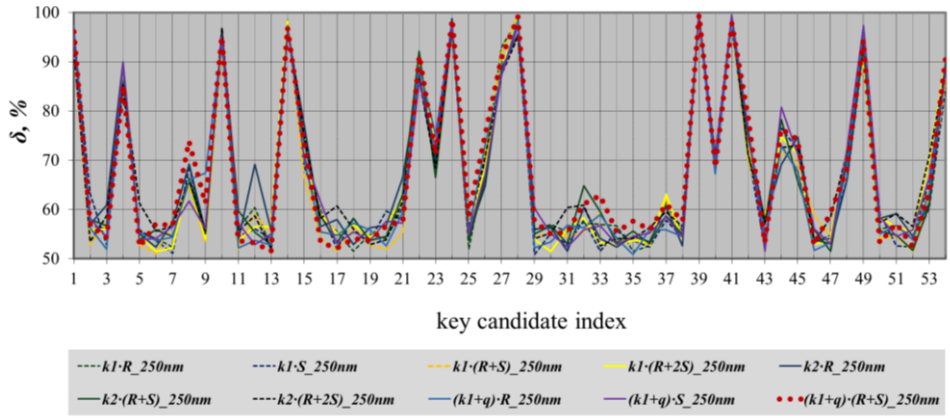


Fig. 22. Correctness of key candidates for all traces analysed in this experiment for the IHP 250 nm technology.

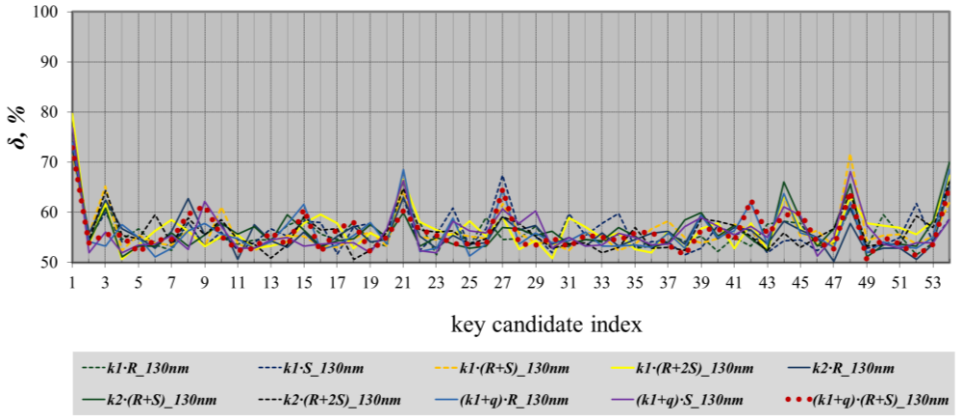


Fig. 23. Correctness of key candidates for all traces analysed in this experiment for the IHP 130 nm technology.

Fig. 22 and Fig. 23 show clearly that:

- the target technology influences the attack resistance of the implemented design significantly;
- the traditional randomization countermeasures do not provide any protection against horizontal SCA attacks.

5.2 GALS-ification

If the regularity principle was not implemented consciously, the implementation can be called a “weak implementation”, i.e. its functionality is correctly implemented but the power profile for the processing a key bit value ‘0’ can be distinguished from the processing a key bit value ‘1’. In this case the knowledge about the implemented algorithm is sufficient to reveal the processed key successfully. Fig. 24 demonstrates this.

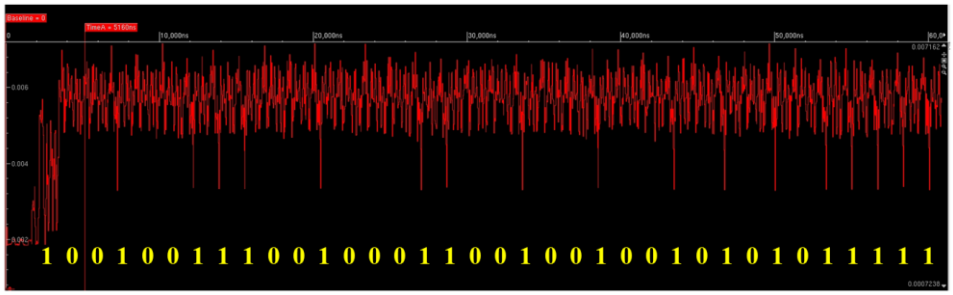


Fig. 24. A part of the simulated power trace of a kP execution of a weak Montgomery kP implementation: the displayed part corresponds to the processing of the first 26 bits of the scalar k.

This design was available at IHP too. Please note, it is not the IHP basic design but one of the previous designs. A power trace for such a weak design was synthesised for the IHP 130 nm technology and for the target clock frequency of about 33 MHz (the clock cycle is 30 ns long). Fig. 24 shows a part of the simulated power trace of a kP execution. This part corresponds to the processing of the first 26 bits of the scalar k. The revealed bits of the scalar k are shown also.

Inspecting the simulated trace the processed scalar k can be easily revealed using the following assumption: the low amplitude – the dip in the trace – corresponds to the processing of a bit value '1'. The dip is not observable when the processing of a key bit value '0' occurs. The explanation of this weakness is easy: the regularity principal was not guaranteed by this Montgomery kP implementation. Please note, that the power trace shown in Fig. 24 was simulated with a time resolution equal to the clock cycle duration, i.e. each clock cycle was represented with only one value that represents the averaged power. Using this compression the dips that are markers for the side channel leakage can be easily detected and used for revealing the key. Fig. 25 demonstrates this. In Fig. 25-*a* the processing of only the first 4 bits of the scalar k is shown: the time resolution of the yellow graph is 0.01 ns, i.e. resulting in 300 power values per clock cycle; the red graph in the middle was simulated with a time resolution of 0.1 ns, i.e. resulting in 30 values per clock cycle and the red graph at the bottom corresponds to the simulation with the time resolution of 30 ns, i.e. only one – the average – value per clock cycle. Fig. 25-*b* shows 3 clock cycles from Fig. 25-*a*, zoomed in.

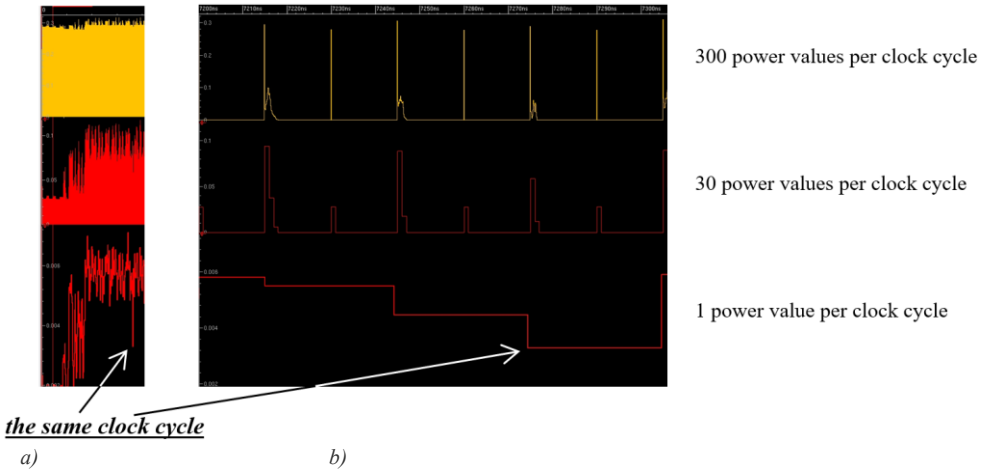


Fig. 25. Power traces of the same kP execution as in Fig. 24 simulated with different time resolutions: 0. 1 ns for the yellow graph; 1 ns for the red graph in the middle; 30 ns for the red graph at the bottom.

Please note that to reveal the key only the knowledge about the implemented algorithm and a reasonable representation of the power trace, that was done using different time resolutions for the trace simulation, was sufficient.

This synchronous IHP ***kP*** design, which was vulnerable to simple SCA attacks, was GALS-ified in IHP, using a pausible clocking scheme, with random hopping of clock frequencies (see details in [67]). The GALS-ification was proposed in [67] as an effective countermeasure against SPA, i.e against horizontal attacks. For the GALS-ification the synchronous design was partitioned into 3 blocks: field multiplier, ALU and block with registers. Each of these blocks was supplied with its own clock. Fig. 26 shows schematically the partitioning of the weak synchronous IHP ***kP*** design into the blocks for the design's GALS-ification, the communication between the blocks and the clock signals of the blocks.

Fig. 26-*c* shows a part of the simulated traces that corresponds to the processing of the synchronous design for 5 clock cycles. Please note that each local clock signal has its own frequency. The frequencies differ significantly from each other, are independent from each other, and are not constant in time. This increases the execution time of a ***kP*** calculation as well as its energy consumption. The GALS-ified design needs about double the time for a ***kP*** execution in comparison to the synchronous design.

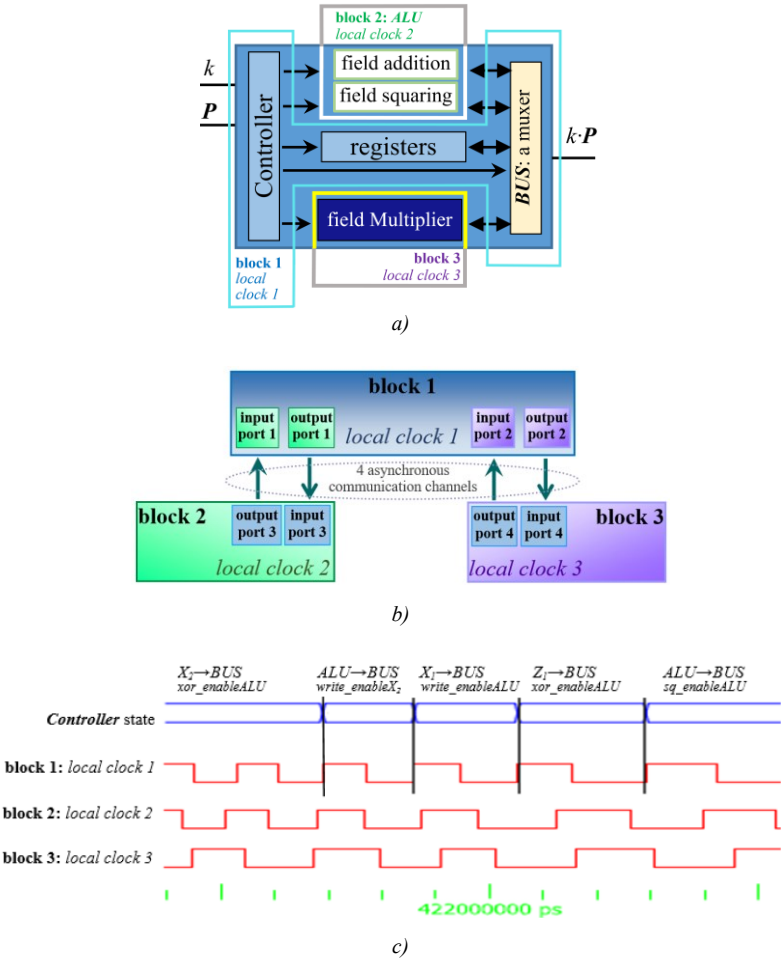


Fig. 26. GALS-ification of the weak synchronous IHP ***kP*** design: (*a*) – partitioning into the blocks for the GALS-ification; (*b*) – communication between the blocks: each block has its own local clock; four asynchronous communication channels were implemented; (*c*) – clock signals for the 3 blocks: part of the simulated trace.

Fig. 27 shows a part of the simulated power traces of the synchronous and the GALS-ified designs, zoomed in. A part at the beginning of the ***kP*** execution is shown. This part corresponds to performing 3 clock cycles of the synchronous design. For the synchronous design the following traces are shown from top to down: the clock signal, the processed bit value of the scalar *k* and the power trace of the whole ***kP*** design. For the GALS-ified design the following traces are shown: the processed value of the scalar *k* (it is '0' in the part of the trace shown), the power traces of three local clock cycle generators and the power trace of the whole ***kP*** design. The clock signal generator of a block consumes energy to switch its local clock signal. At the end of the generator activity the local clock signal is switched and activates the block, i.e. the block starts to consume the energy. The simulations were done with the simulation step Δt=1 ns, i.e. for each clock cycle of the synchronous design 30 power values are recorded.

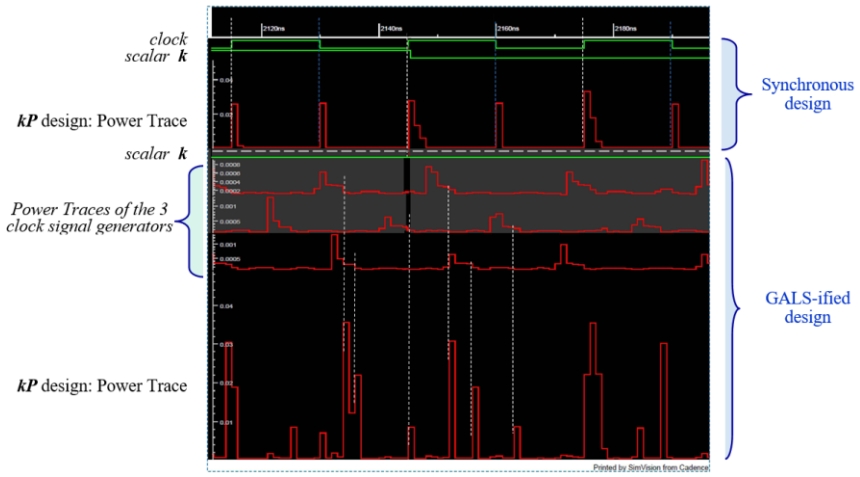


Fig. 27. Simulated power traces of the synchronous and the GALS-ified designs, zoomed in. A part at the beginning of the ***kP*** execution is shown. This part corresponds to executing 3 clock cycles of the synchronous design. For the GALS-ified design four power traces are shown: the power traces of the three local clock cycle generators and the power trace of the whole ***kP*** design.

Fig. 27 shows clearly that the energy consumption of the GALS-ified ***kP*** design is significantly higher than the one of the synchronous design within the same time interval. At the first look the power shape of the GALS-ified design seems to be more complicated than the one of the synchronous design. This can lead to the assumption that the GALS-ification makes the design more resistant against simple SCA attacks, but it is only the first impression that is not true in reality. Fig. 28 shows a part of the PT of the ***kP*** execution simulated with a time resolution of 0.1 ns. In this case the activity of the block with the highest energy consumption can be clearly seen. The fact that the most consuming block in a ***kP*** design is the filed multiplier is well-known. Thus, if an attacker can see the trace that corresponds to the activity of only the field multiplier within the whole ***kP*** operation, he can concentrate on attacks specialized on exploiting different multiplier weaknesses. The same is also true for other blocks, for example the block with the registers. Thus, the GALS-ification implemented in [67] can even help an attacker to perform a successful analysis of the activity of registers. Fig. 28 demonstrates how the common knowledge about the power consumption of the mathematical operations and registers allows to understand, what we "see" in the PT. The letter **M** marks the peak that corresponds to the activity of the field multiplier, which is the biggest and most energy-consuming block of the design, i.e. it is *block 3* of the GALS-ified ***kP*** design. It is more complicated to understand, which peaks correspond to the activity of the block with the registers (see *block 1* in Fig. 26) and of the ***ALU*** (see *block 2* in Fig. 26). The common knowledge that can help to select the activity of the registers is the fact that their activity is very short in time. In Fig. 28 letter ***R*** denotes the peak that corresponds to the activity of the block with

registers and letter ***A*** corresponds to the activity of the ***ALU***. We did this separation using the knowledge of designers about the activity of the clock signal generators, i.e. comparing PTs of clock generators with PTs of the design blocks. But the short pulse duration allows to distinguish the power profile of the ***ALU*** from the one of the registers, even for attackers.

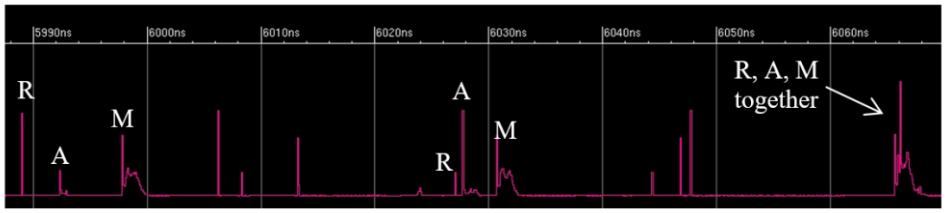


Fig. 28. A part of the power trace of the ***kP*** execution of the GALS-ified design simulated with Δt=0.1 ns. The power profiles of the three design blocks differ significantly from each other and can be distinguished using common knowledge about the implementation details of mathematical operations.

Please note that for the GALS-ification described in [67] the following is true: each pulse in the power trace of the synchronous design corresponds to a "triplet" in the power trace of the GALS-ified design. Moreover, the pulse that corresponds to the multiplier can be clearly identified in each triplet. Due to this fact the distinguisher that we used for the successful SPA on the synchronous design has to be successful also when attacking the GALS-ified design. This means that each clock cycle with low power in the synchronous design (that is the distinguishing marker for the processing of a key bit value '1') has to correspond to a triplet where usually the biggest power consumption is now significantly lower than in other triplets. Thus the key can be easily revealed attacking the GALS-ified design too. Fig. 29 demonstrates this. The green line is a part of the PT of the synchronous weak IHP design. The blue traces is a part of the PT of the GALS-ified design. The processed key bit values for the synchronous design are shown at the top of the diagram, see green marked sequence of '1' and '0'. The processed key bit values for the GALS-ified design are shown at the bottom of the diagram, see magenta marked sequence of '1' and '0'. The parts of the traces marked with white dashed rectangles in Fig. 29 correspond to the processing of the same key bit of the processed scalar *k*.

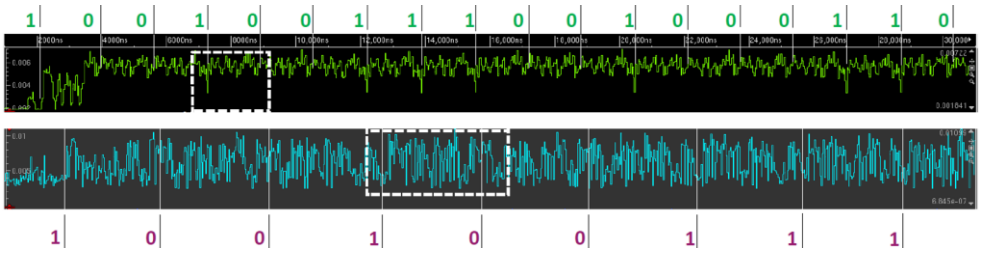


Fig. 29. A part of power traces of the ***kP*** execution of the synchronous (top) and the GALS-ified (bottom) designs.

Due to the fact that the distinguishing marker is the low energy consumption during a clock cycle, it can be easy detected in the green trace, especially when a simulation step of Δt=30 ns is applied, i.e. each dip in the green trace corresponds to the processing of a key bit '1'. In case of the GALS-ified design the "coarse" simulation time step makes interpreting the trace complicated, but even now the same distinguishing marker, i.e. the low energy consumption at the end of the processing of each key bit value '1', can be detected in the blue line. Analysis of the trace simulated with the time step of Δt=1 ns is much easier, see Fig. 30. The violet line in Fig. 30 corresponds to the part of the blue trace identified by the rectangle in Fig. 29, zoomed in.

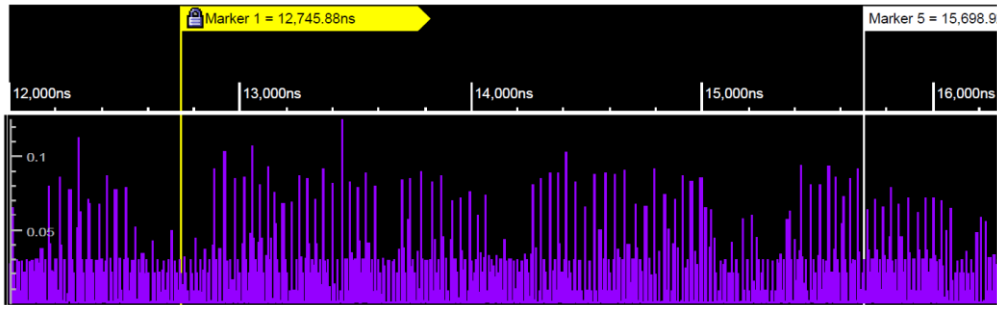


Fig. 30. A part of power traces of the ***kP*** execution of the whole GALS-ified design.

Two time markers show the end of the time slots for the processing of a key bit. The yellow time marker corresponds to the end of the processing of a key bit value '1'. The white time marker corresponds to the end of the processing of a key bit value '0'. The fact that the design consumes much less energy at the end of the processing a key bit '1' can successfully be exploited for revealing the key also for the GALS-ified design. At the end of the processing of a '1' (see yellow time marker) idle state of the multiplier between two field multiplication is long, i.e. we see a gap between groups with high peaks. At the end of the processing of a '0' the field multiplier doesn't idle, i.e. we see the white time marker in the middle of a group with high peaks. Using this "new" distinguisher the key can be revealed successfully even for the trace simulated with the coarse simulation step of 30 ns, see Fig. 29. This shows clearly that a straightforward GALS-ification of a synchronous ECC design that is vulnerable to SPA leads to a design that is also vulnerable to SPA, i.e. my results cannot confirm the results published in [67].

GALS-ification can be a good countermeasure, but the GALS-ification may not be straightforward, i.e. it needs to be more sophisticated than just using existing blocks. Otherwise, the weaknesses of a synchronous design will survive the transformation leading to the fact that the GALS-ified design is also vulnerable [113]. This is due to the fact that in a GALS-ified design the activity of the single blocks can be observable. This makes the GALS-ified designs even more vulnerable to a wide spectrum of SCA attacks. When a GALS-ified design is analysed in order to verify its vulnerability a reasonable representation of the simulation results and even more important a proper definition of the simulations steps is of utmost importance. Otherwise, the results may lead to misinterpretations, especially to ones emphasising resistance against SCA.

In order to really improve the SCA resistance of a design using GALS-ification a sophisticated strategy is paramount. A potential approach is splitting the design into a lot of blocks with similar power shapes, so that the power profiles of different operations are no longer distinguishable from each other.

5.3 Breaking a fully Balanced ASIC Coprocessor Implementing Complete Addition Formulas on Weierstrass Elliptic Curves

In this section we report on the results of horizontal SCA attacks against open-source designs that implement hardware accelerators for the ***kP*** operation for the Elliptic Curve *secp256k1* [114]. The design implements the Montgomery ladder algorithm with a randomized sequence of the EC point doublings and additions. This kind of randomization can be a good countermeasure against different SCA attacks if power shapes of a point doubling and a point addition are indistinguishable. For this reason, both EC point operations have to be implemented (at least) using a universal EC point addition formula.

A fully balanced ASIC Coprocessor implementing complete addition formulas on Weierstrass Elliptic Curves was reported in [63], whereby the EC point doublings as well the point additions were implemented corresponding to a universalized point addition formula. Additionally, the sequence of EC point operations in the main loop of the Montgomery ladder was randomized as a countermeasure against vertical SCA attacks. The design is open-source and implementation details are given in [115].

In this section the randomized Montgomery $k\boldsymbol{P}$ design for the EC *secp256k1* proposed in [63] is investigated. The design implements a random order execution according to the algorithm proposed in [116]. We give the algorithm in Appendix 2. The code is available through GitHub [117]. To the best of our knowledge, the resistance of the design against attacks was not evaluated in [63] or any other publication. We used *the comparison to the mean* to attack the design and we were able to extract the processed keys with a correctness of 100%. We published our results in [118].

The design that executes the point operations in a randomized order in each iteration of the Montgomery ladder is considered to be more resistant against SCA attacks, especially against vertical DPA attacks which mainly target storing values in registers. We denote the investigated design as a *design with countermeasures* in the rest of this section. Inputs of the design are: the scalar m; X, Y and Z coordinates of the EC point P, a set of control signals (*rst, clk, ce, load*) and a random number r which defines the random execution order for the EC point doublings and additions.

We synthesized the *design with countermeasures* for the IHP 250 nm cell library SGB25V [119] using the Synopsys Design Compiler Version K-2015.06-SP2. The maximum achieved clock frequency is 23.256 MHz (43 ns clock cycle period).

In our experiments we did not use the 20-bit long scalar m = 0xbe9ff that was used in the original testbench as it is pretty short and predominantly consists of ones. Instead, we used a 252-bit long random scalar:

$$m=0x9be627ea91dc5bbac55a06295ce870b07029bfcdb2dce28d959f2815b16f817.$$

We used the following scalar r to ensure a random execution order:

$$r=3746cb5ed29e53453b0ff49f78e88bea61d8de75b8f55ab9a112d06bad0afc9.$$

We used the basis point of EC *secp256k1* [114] as the input point $\boldsymbol{P}$ corresponding to the original testbench.

For our experiments we used a design synthesized for a frequency of 20 MHz. It requires about 114 ms for processing a 252-bit long scalar m. Other design parameters are discussed in [118]. In this section we concentrate on the resistance of the design against horizontal attacks only.

For the functional verification of the design, we used Cadence SimVision 15.20-s053. A part of the simulations of the *design with countermeasures* is shown in Fig. 31. It can be seen that parameter r_i (see signals *r_t* in Fig. 31) determines the sequence of point operations, see signals *state* in Fig. 31 after initialization (marked with time flag TimeA). For r_i = '0' a point addition (*state=s_add*) is followed by point doubling (*state=s_double*). For r_i = '1' a point doubling (see time flag TimeB, *state=s_double*) is followed by a point addition (*state=s_add*). The time for the processing of a single bit of the scalar m can be also seen in our simulation in Fig. 31. It is the time between time flags TimeA and TimeB. Processing of a single bit of the scalar m takes TimeB-TimeA=*677675 ns – 226325 ns = 451350 ns*. It corresponds to 9027 clock cycles for a clock cycle period of 50 ns.

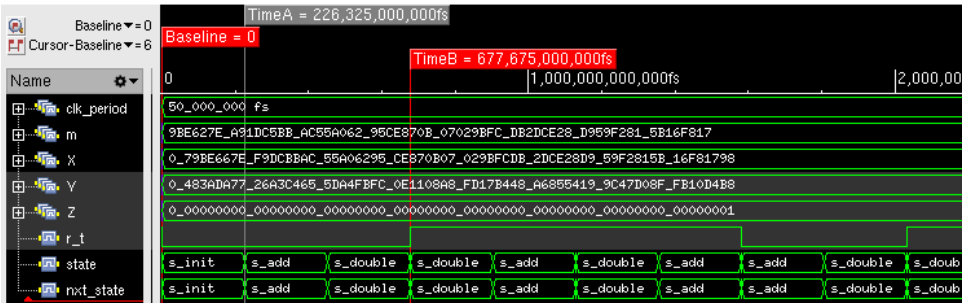


Fig. 31. Beginning of the simulation for the *design with countermeasures* demonstrating a sequence of point addition and point doubling operations depending on parameter r_i (see algorithm in Appendix 2).

In order to test the resistance of the design, we generated power traces of the ***kP*** execution using Synopsys PrimeTime (R) Version Q-2019.12-SP1. We used the comparison to the mean (see section 3.2.3) for the statistical analysis of the trace simulated for the top module (*point_mult*) during the execution of the *mP* operation. Due to the fact that simulated traces are noise-free, we set the time step in our simulation equal to the period of the clock cycle. Thus, each clock cycle in our simulation was represented only by 1 sample. We decided to do so to avoid a huge memory consumption as the processing of a single key bit in the algorithm's main loop takes 9027 clock cycles, i.e. the complete trace is about 2.3 Mio. clock cycles long. The file in which the simulation results for a single top block are stored is about 0.5 gigabyte even when a coarse time step for the simulation is used. In our attack we extracted 1 key candidate per clock cycle, i.e. we obtained 9027 key candidates. The success of our attack represented as the correctness of each of the key candidates is shown in Fig. 32. The correctness of the key candidate derived from the first clock cycle and clock cycle 4514 is about 75.5% and 79.6% respectively.

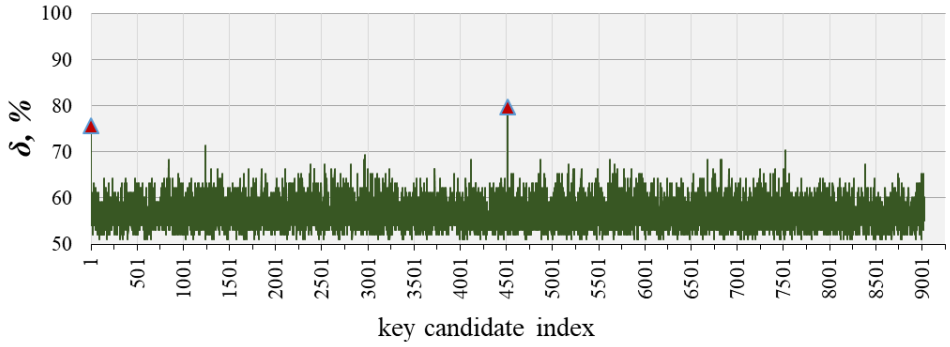


Fig. 32. Results of our horizontal attack using *comparison to the mean* against the *design with countermeasures*.

Due to the relatively high success rate of the attack performed using the compressed trace and to the fact that the compressing itself influences the attack results (i.e. it can "hide" SCA leakages, or – opposite – can make them more "visible"), we decided to perform the simulation a second time, with 100 samples per clock cycle. The file in which the simulation results are stored is about 48 gigabytes. The automated SPA attack described in section 3.2.5.1 allows easy to detect the observable differences in shapes of slots corresponding to the processing of different key bit values. Our program helps to detect very small differences in a similar way as a magnifying glass would do. We applied the automated SPA attack to the *design with countermeasures*.

Results of attacking the top block are presented in Fig. 33. The correctness of the key revealed at the beginning of the slot is only 78.6% (see Fig. 33-*b*) but the leakage detected at

points in the middle of the slot – samples 451351 and 451352, see Fig. 33-*c* – allows now to reveal the processed scalar completely.

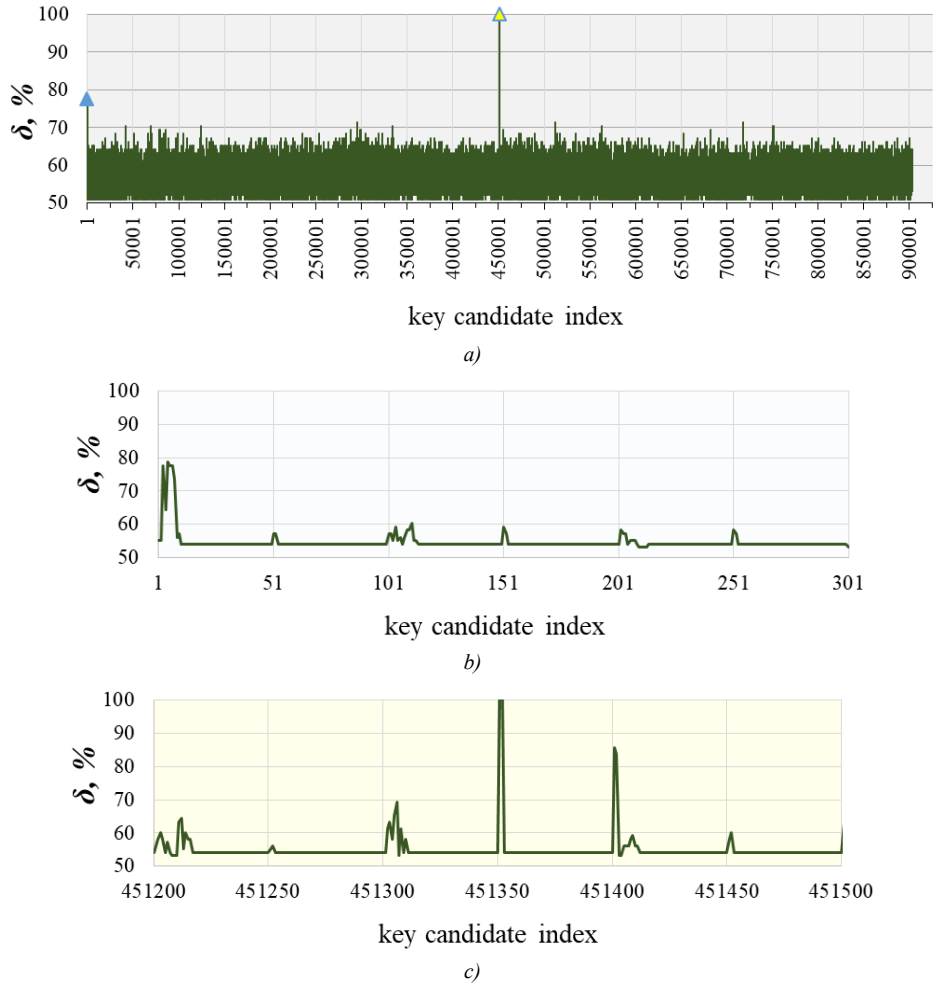


Fig. 33. Results of the sample-wise horizontal DPA attack against the *design with countermeasures (a)* and selected parts of the trace corresponding to the key candidates with a high correctness, zoomed in *(b)*, *(c)*.

We used the sample 451352 of each slot, that corresponds to a strong leakage source, to visualize our automated SPA attack. Fig. 34 shows the power consumed at that point in time during the processing of the first 50 bits of the scalar "9BE627EA91DC5" except of its most significant bit. It can be clearly seen if a bit equal to '0' is processed the power consumed is higher than 1.2 mW.

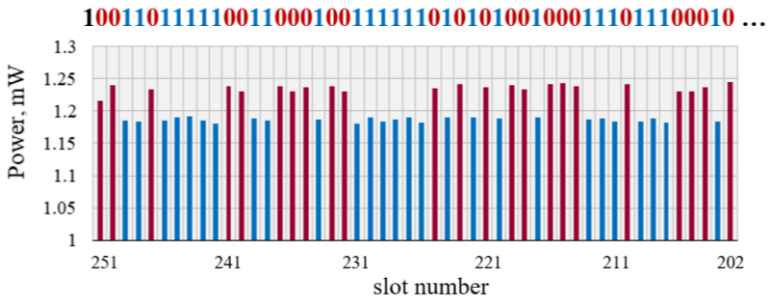


Fig. 34. Values of the power consumed in slot's sample 451352 extracted from each of the first 50 slots of the simulated power trace of the *point_mult* block.

In this section we described our successful attack against a randomized Montgomery ladder with indistinguishable EC point doubling and point addition operations implemented using a universal EC point addition formula. Despite the fact that we are able to reveal the processed scalar completely analysing the simulated traces, we doubt that the attack can be successfully conducted against a measured trace. One reason is related to the numerous technical difficulties accompanying the measurements of a very long trace with a sufficiently high sampling rate. Another reason is that the measured traces are not noiseless. These factors reduce the "visibility" of the SCA leakages. Due to these facts, the randomization strategy can be a promising countermeasure. But its disadvantage is the long execution time caused by many field multiplications that are necessary for the implementation of the main loop with the unified EC point addition formula (28 field multiplications per loop, see [63]). Thus, we decided not to pursue this strategy any further.

6 Improvement of the IHP basic *kP* design

In order to improve the resistance of our basic design we investigated its major leakage sources as well as several well-known and often used countermeasures. In section 4 we determined the ***BUS*** as the main SCA leakage source. In section 5 it was shown that the well-known countermeasures, mostly based on randomization of the processed data, the clock frequency, or even the sequence of mathematical operations are not effective against horizontal attacks.

Introducing noise is a well-known countermeasure against simple SCA attacks and can be an efficient alternative strategy to increase a design's resistance. The multiplier is a big block of the ***kP*** design. Its energy consumption is high in comparison to other design's blocks. Thus, the activity of the multiplier is a kind of noise that can hide the activity of other blocks, i.e. the activity of the ***BUS*** can be (at least partially) hidden by the field multiplier. Please note that it is necessary to verify first that the field multiplier is resistant against the horizontal attacks and is not an SCA leakage source itself.

In this work we did not investigate if the countermeasures against vertical address-bit DPA [45], [46] (see section 2.3.2) are effective against horizontal address-bit DPA too. We concentrated on methods increasing the inherent resistance of the design. In this section we investigate the hiding effect of multipliers implemented using different multiplication formulae and propose a regular schedule for addressing the blocks of a design to reduce the vulnerability of the ***BUS,*** i.e. the major leakage source in our design.

6.1 Multiplier as a means for hiding SCA leakages

The investigated field multiplier is a serial implementation of the 4-segment iterative Karatsuba MM that requires a calculation of 9 partial products. Each partial product is calculated in the block Partial Multiplier (PartMult) of the field multiplier. The structure of the partial multiplier as well as the calculation sequence of partial products can influence the resistance of the whole ***kP*** design [120]-[121]. In subsection 6.1.1 we investigate the influence of different multiplication formulae applied for the implementation of the partial multiplier on the resistance of the whole ECC design. In subsection 6.1.2 we investigate the impact of the randomized sequence for partial product calculations on the design resistance. We discuss the hiding features of the multiplier in section 6.1.3.

6.1.1 Influence of different multiplication methods

Different multiplication formulae being implemented in hardware result in different circuits, containing a different number of gates. This causes a different energy consumption for the processing of the same operands, i.e. for a calculation of the same partial product. Thus, the hiding effect of the field multiplier depends on the multiplication formula used for the implementation of its partial multiplier.

We start with a description of the partial multiplier of the basic IHP ***kP*** design. In order to investigate the success of horizontal attacks depending on the MM applied for the calculation of the partial product we implemented 4 additional designs. All designs differ n only in their Partial Multiplier from the basic IHP ***kP*** design. We explain the implementation details of each of the investigated partial multipliers and discuss the results of our attacks in the following subsections. We demonstrate that the classical multiplication formula applied to implement the partial multiplier can increase the inherent resistance of the whole ECC design.

6.1.1.1 IHP basic kP design: structure of the field multiplier and its partial multiplier

Fig. 35 shows the structure of the field multiplier in the basic IHP ***kP*** design detailed. In each of the 9 clock cycles of a field multiplication one partial polynomial product of two 59-bit long operands A^j and B^j (with $1 \le j \le 9$) is calculated and accumulated to the field product including its reduction.

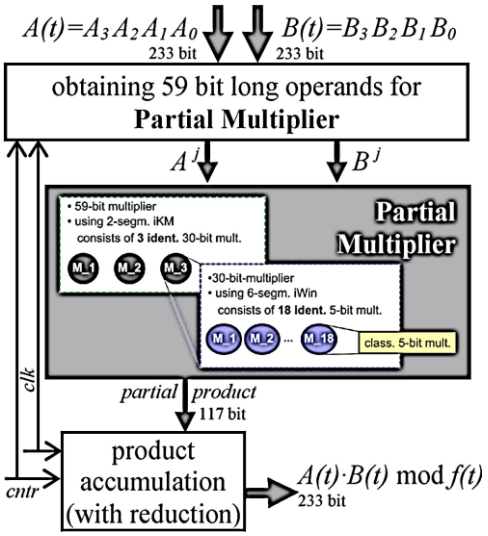


Fig. 35. Structure of the field multiplier in the basic IHP ***kP*** design.

The block partial multiplier of the IHP basic ***kP*** design is implemented as a combination of the 3 multiplication methods: the 2-segment Karatsuba multiplication formula [122], the 6-segment iterative Winograd MM [123] and the classical MM [1] for 5-bit long operands.

The 2-segment Karatsuba multiplication formula was in the IHP basic ***kP*** design iteratively applied to 60-bit long multiplicands [123].

$$\begin{aligned} A^j \cdot B^j &= \left(A^j{}_1 \cdot 2^m \oplus A^j{}_0\right) \cdot \left(B^j{}_1 \cdot 2^m \oplus B^j{}_0\right) = \\ &= \underbrace{A^j{}_1 B^j{}_1}_{\substack{2m-1 \text{ bit long} \\ \text{result}}} \cdot 2^{2m} \oplus \underbrace{\left(\left(A^j{}_1 \oplus A^j{}_0\right)\left(B^j{}_1 \oplus B^j{}_0\right) \oplus A^j{}_1 B^j{}_1 \oplus A^j{}_0 B^j{}_0\right)}_{2m-1 \text{ bit long result}} \cdot 2^m \oplus \underbrace{A^j{}_0 B^j{}_0}_{\substack{2m-1 \text{ bit long} \\ \text{result}}} - \\ &= \underbrace{C_3 C_2 C_1 C_0}_{4m-1 \text{ bit long result}} \end{aligned} \tag{30}$$

The gate complexity (GC) of this multiplier is $GC_{2m}=3 \cdot GC_m+(7m-3)_{XOR}$. Here m is the length of segments $m=60/2=30$ and GC_m is the gate complexity of the internal m-bit partial multipliers. The symbol '$\oplus$' denotes the addition of the $GF(2^n)$ elements that is a bitwise XOR operation. The multiplication of a number with 2^i means the bitwise left shift of the operand by i bits.

Thus, the 59-bit partial multiplier consists of 3 internal multipliers: two of them for 30-bit long operands (for the multiplications $A^j{}_0 B^j{}_0$ and $\left(A^j{}_0 \oplus A^j{}_1\right)\left(B^j{}_0 \oplus B^j{}_1\right)$) and one multiplier for 29-bit long operands (for the multiplication $A^j{}_1 B^j{}_1$). All these internal multipliers are implemented identically, using the following 6-segment iterative Winograd multiplication formula derived corresponding to the description given in [123]:

$$
\begin{aligned}
a\cdot b = a_5a_4a_3a_2a_1a_0\cdot b_5b_4b_3b_2b_1b_0 = \sum_{l=0}^{5} a_l\cdot 2^{l\cdot m} = \underbrace{a_5b_5}_{\substack{2m-1 \text{ bit long}\\ \text{result}}}\cdot 2^{10m}\\
\oplus\underbrace{(a_5b_4\oplus a_4b_5\oplus(a_5\oplus a_4)(b_5\oplus b_4))}_{2m-1 \text{ bit long result}}\cdot 2^{9m}\oplus\underbrace{(a_5b_3\oplus a_4b_4\oplus a_3b_5\oplus(a_5\oplus a_4)(b_5\oplus b_4))}_{2m-1 \text{ bit long result}}\cdot 2^{8m}\\
\oplus\underbrace{(a_5b_2\oplus a_4b_3\oplus a_3b_4\oplus a_2b_5\oplus(a_5\oplus a_4)(b_5\oplus b_4)\oplus(a_5\oplus a_4)(b_5\oplus b_4))}_{2m-1 \text{ bit long result}}\cdot 2^{7m}\\
\oplus\underbrace{\left(\begin{array}{l}a_1b_1\oplus a_2b_2\oplus a_3b_3\oplus a_4b_4\oplus a_5b_5\oplus(a_1\oplus a_4)(b_1\oplus b_4)\oplus(a_2\oplus a_5)(b_2\oplus b_5)\oplus\\(a_5\oplus a_4)(b_5\oplus b_4)\oplus(a_5\oplus a_4)(b_5\oplus b_4)\oplus(a_5\oplus a_4\oplus a_5\oplus a_4)(b_5\oplus b_4\oplus b_5\oplus b_4)\end{array}\right)}_{2m-1 \text{ bit long result}}\cdot 2^{6m}\\
\oplus\underbrace{\left(\begin{array}{l}a_0b_0\oplus a_1b_1\oplus a_2b_2\oplus a_3b_3\oplus a_4b_4\oplus a_5b_5\oplus(a_0\oplus a_3)(b_0\oplus b_3)\oplus(a_1\oplus a_4)(b_1\oplus b_4)\oplus\\(a_2\oplus a_5)(b_2\oplus b_5)\oplus(a_0\oplus a_2)(b_0\oplus b_2)\oplus(a_3\oplus a_5)(b_3\oplus b_5)\oplus\\(a_5\oplus a_4\oplus a_5\oplus a_4)(b_5\oplus b_4\oplus b_5\oplus b_4)\end{array}\right)}_{2m-1 \text{ bit long result}}\cdot 2^{5m}\\
\oplus\underbrace{\left(\begin{array}{l}a_0b_0\oplus a_1b_1\oplus a_2b_2\oplus a_3b_3\oplus a_4b_4\oplus(a_0\oplus a_3)(b_0\oplus b_3)\oplus(a_1\oplus a_4)(b_1\oplus b_4)\oplus\\(a_2\oplus a_5)(b_2\oplus b_5)\oplus(a_0\oplus a_2)(b_0\oplus b_2)\oplus(a_3\oplus a_5)(b_3\oplus b_5)\oplus\\(a_5\oplus a_4\oplus a_5\oplus a_4)(b_5\oplus b_4\oplus b_5\oplus b_4)\end{array}\right)}_{2m-1 \text{ bit long result}}\cdot 2^{4m}\\
\oplus\underbrace{(a_4b_4\oplus a_3b_4\oplus a_4b_3\oplus a_2b_4\oplus a_3b_3\oplus a_4b_2\oplus(a_4\oplus a_3)(b_4\oplus b_3)\oplus(a_4\oplus a_3)(b_4\oplus b_3))}_{2m-1 \text{ bit long result}}\cdot 2^{3m}\\
\oplus\underbrace{(a_4b_3\oplus a_3b_4\oplus a_3b_3\oplus(a_4\oplus a_3)(b_4\oplus b_3))}_{2m-1 \text{ bit long result}}\cdot 2^{2m}\oplus\underbrace{(a_4b_4\oplus a_3b_3\oplus(a_4\oplus a_3)(b_4\oplus b_3))}_{2m-1 \text{ bit long result}}\cdot 2^{m}\\
\oplus\underbrace{a_0b_0}_{\substack{2m-1 \text{ bit long}\\ \text{result}}}
\end{aligned}
\tag{31}
$$

The gate complexity corresponding to (31) is $GC_{6m}=18\cdot GC_m+(72m-19)_{XOR}$, with $m=30/6=5$ bits. The GC_m is the gate complexity of the internal m-bit partial multipliers, i.e. the multipliers for 5-bit long operands.

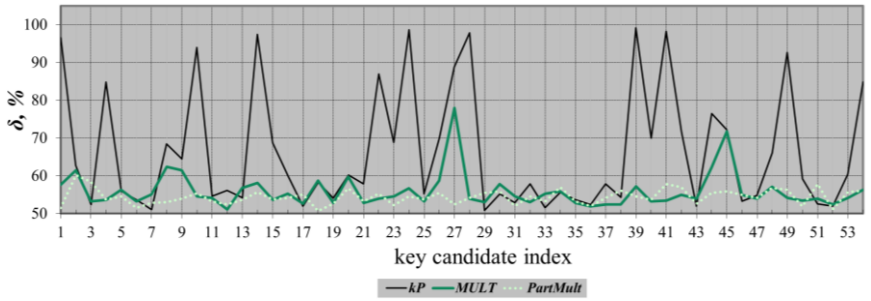
Thus, each partial multiplier for 30-bit long operands consists of 18 internal multipliers for 5-bit long operands $a=a_4a_3a_2a_1a_0$ and $b=b_4b_3b_2b_1b_0$. Each of these small multipliers was implemented using the classical multiplication formula with $m=5$:

$$c = a\cdot b = \sum_{i=0}^{2m-2} c_i\cdot 2^i, \quad \text{with} \quad c_i = \bigoplus_{i=k+l} a_k\cdot b_l, \quad \forall k,l<m \tag{32}$$

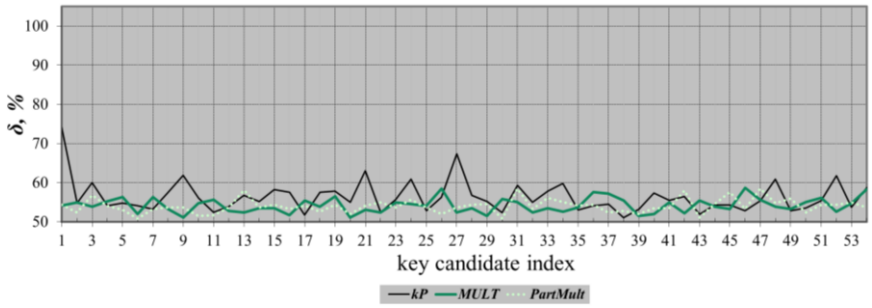
The gate complexity of each of the m-bit classical multipliers corresponding to formula (32) is $GC_m=m^2_{\&}+(m-1)^2_{XOR}$, *i.e.* $5^2=25$ *AND* gates and $4^2=16$ *XOR* gates for m=5.

Applying the combination of 3 multiplication methods described above, the gate complexity of the 59-bit partial multiplier is $GC_{59}=1350_{\&}+2094_{XOR}$. We denote this partial multiplier as ***basic*** (and the IHP basic ***kP*** design as *design1* in the rest of this section).

The resistance of the ***basic*** partial multiplier as well as the one of the field multiplier against horizontal attacks was evaluated for the IHP basic ***kP*** design implemented in the IHP 250 nm and in the IHP 130 nm technologies. We used the *comparison to the mean* approach for the statistical analysis of simulated power traces (see section 3.2.3). The traces were compressed: we used the average power value of a clock cycle to represent the clock cycle. Fig. 36 represents the attack results.



a) 250 nm technology



b) 130 nm technology

Fig. 36. Success rate attacking the *design1* (i.e. the IHP basic ***kP*** design) synthesised for both IHP technologies: traces of whole ***kP*** design, its field multiplier (**MULT**), and the *basic* partial multiplier (PartMult).

The field multiplier in IHP basic design synthesised for the IHP 130 nm technology is resistant against the performed horizontal power analysis attack. The resistance of the design obtained using the same VHDL code synthesised for the IHP 250 nm technology is smaller than the one of the 130 nm technology. The resistance of the partial multiplier is high in both cases: the correctness δ of all key candidates is in the range of 50% and 60%.

Additionally, the resistance of the field multiplier in the IHP basic ***kP*** design against Horizontal Collision Correlation Analysis attacks introduced in [20] was investigated. We showed in [43] that the multiplier is resistant against collision attacks. This is achieved by using the iterative 4-segment Karatsuba MM:

- the number of calculated partial products is small (only 9);
- the length of operands for each partial multiplication is big, w=59;
- operands for partial product calculation using the iterative Karatsuba MM are different, that is not the case if the classical MM as assumed in [20] is applied.

More details about the analysis can be found in [43].

6.1.1.2 Implemented partial multipliers

In this subsection we explain details of the partial multipliers discussed above.

Partial multiplier using the classical MM

This partial multiplier denoted further as ***clas*** was implemented using only the classical multiplication formula, i.e. it implements formula (32) with m=59.

The gate complexity of this multiplier, i.e. the amount of AND and XOR gates which are necessary to implement its functionality corresponding to formula (32) is m^2 AND gates and $(m-1)^2$ XOR gates, i.e.: $GCm_{=59}=3481_{\&}+3364_{XOR}$.

In the rest of this thesis we denote the IHP basic ***kP*** design with the partial multiplier implementing the classical MM as *design2*.

Partial multiplier using the iterative 4-segment Karatsuba MM and the classical MM

This partial multiplier (further denoted as ***ikm***) processing 59-bit long operands was implemented using the 4-segment iterative Karatsuba multiplication formula (see [77], [123]-[124]) applied for 60-bit long multiplicands A and B:

$$\begin{aligned} A\cdot B &= A_3A_2A_1A_0\cdot B_3B_2B_1B_0 = \\ &= A_0B_0\cdot 2^0 \oplus \left(A_0B_0 \oplus A_1B_1 \oplus (A_0\oplus A_1)(B_0\oplus B_1)\right)\cdot 2^m \\ &\quad \oplus \left(A_0B_0 \oplus A_1B_1 \oplus A_2B_2 \oplus (A_0\oplus A_2)(B_0\oplus B_2)\right)\cdot 2^{2m} \\ &\quad \oplus \left(\begin{array}{l} A_0B_0 \oplus A_1B_1 \oplus A_2B_2 \oplus A_3B_3 \oplus (A_0\oplus A_2)(B_0\oplus B_2) \\ (A_0\oplus A_1)(B_0\oplus B_1)\oplus (A_1\oplus A_3)(B_1\oplus B_3)\oplus (A_2\oplus A_3)(B_2\oplus B_3)\oplus \\ (A_0\oplus A_1\oplus A_2\oplus A_3)(B_0\oplus B_1\oplus B_2\oplus B_3) \end{array}\right)\cdot 2^{3m} \\ &\quad \oplus \left(A_1B_1 \oplus A_2B_2 \oplus A_3B_3 \oplus (A_1\oplus A_3)(B_1\oplus B_3)\right)\cdot 2^{4m} \\ &\quad \oplus \left(A_2B_2 \oplus A_3B_3 \oplus (A_2\oplus A_3)(B_2\oplus B_3)\right)\cdot 2^{5m} \oplus A_3B_3\cdot 2^{6m} \end{aligned} \tag{33}$$

The gate complexity of this multiplier is $GC_{4m}=9\cdot GC_m+(34m-11)_{XOR}$. Here m is the length of the segments $m=60/4=15$ and GC_m is the gate complexity of the internal m-*bit* partial multipliers. In this partial multiplier we implemented all 9 internal 15-bit partial multipliers identically. Each of them was again implemented using the 4-segment iterative Karatsuba multiplication formula. Thus, the 15-bit multiplier consists of 9 internal 4-bit multipliers, each implemented using the classical multiplication formula (32) with $m=4$. Fig. 37 shows the structure of ***ikm***.

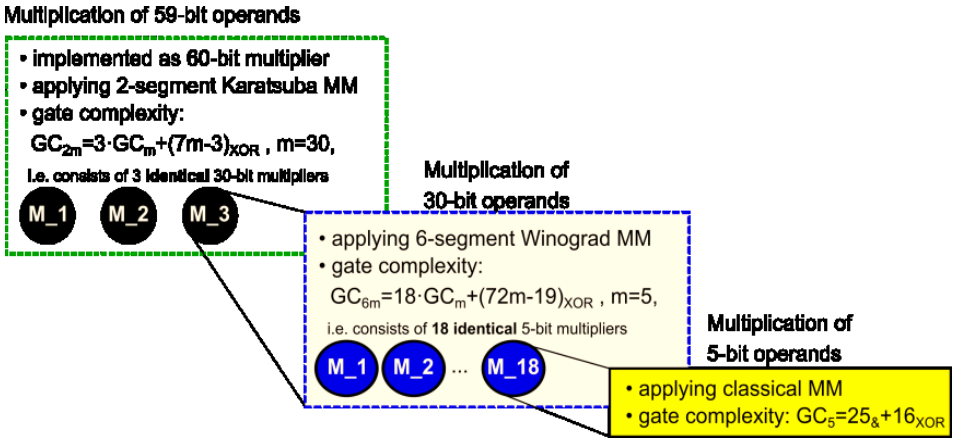


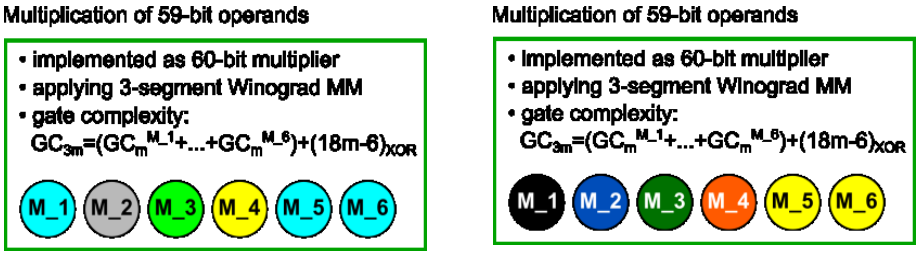
Fig. 37. Structure of the partial multiplier ***ikm***.

In the following the design with the ***ikm*** partial multiplier is denoted as *design3*.

*Partial multipliers **combi1** and **combi2***

These partial multipliers of 59-bit long operands were implemented as different combinations of 3 multiplication formulae: the 3-segment iterative Winograd MM, the 4-segment iterative Karatsuba MM and the classical MM. At first the 3-segment iterative Winograd MM

was applied for 60-bit long operands. This MM defines the number of the internal multipliers, ***combi1*** and ***combi2*** contain 6 internal multipliers *M_1,…,M_6* for 20-bit long operands. The multipliers *M_1*, *M_5* and *M_6* in ***combi1*** are identical (see blue marked multipliers in Fig. 38-*a*. The multiplier *M_5* and *M_6* in ***combi2*** are identical to the multiplier *M_1* in ***combi1*** and are implemented using the classical MM only (see yellow marked multipliers in Fig. 38-*a* and Fig. 38-*b*). The combination of MMs for the implementation of the multipliers *M_1,…,M_4* in ***combi1*** and all multipliers of ***combi2*** was chosen randomly.



*a) - **combi1*** *b) – **combi2***

Fig. 38. Partial multipliers used to realize ***combi1*** and ***combi2***.

We do not give details here for the gate complexity of the partial multipliers ***combi1*** and ***combi2***. Important is only the fact, that the complexity of both partial multipliers as well as of the ***clas*** and the ***ikm*** partial multipliers as well as the partial multiplier in the IHP basic ***kP*** design is different.

The design with the ***combi1*** partial multiplier we denote here as *design4*; the design with the ***combi2*** partial multiplier we denote here as *design5*.

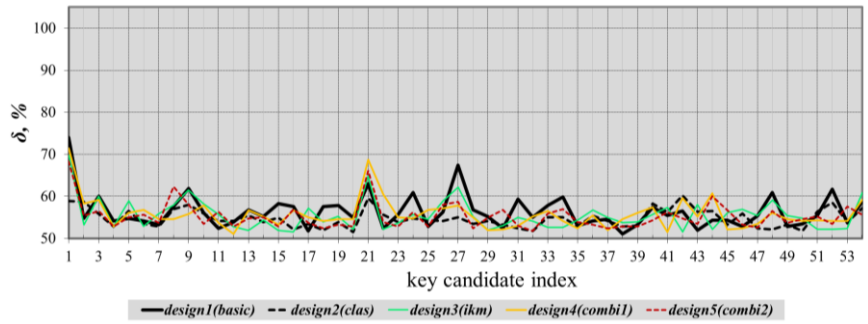
6.1.1.3 Analysis results

We synthesized the ***kP*** designs using the Partial Multipliers described here for the IHP 130 nm and 250 nm technologies. TABLE VI. gives a short overview of the parameters of the ***kP*** designs with the different partial multipliers described here. The goal is to show that the main parameters of the investigated ***kP*** designs such as their area and power are different.

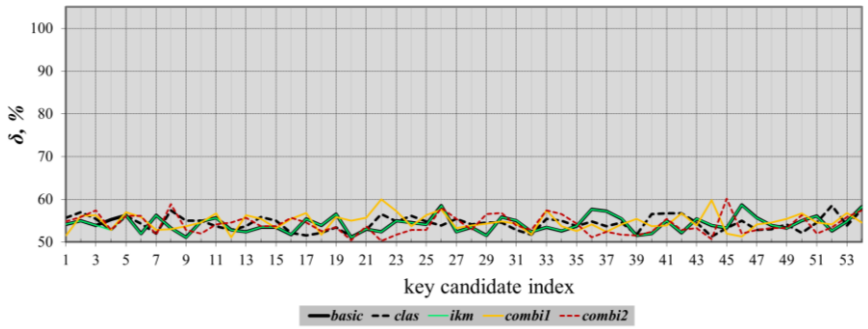
TABLE VI. PARAMETERS OF *KP* DESIGNS WITH DIFFERENT PARTIAL MULTIPLIERS

Technology	Investigated *kP* designs			Field multiplier				
					area		power	
	name	area, mm^2	power mW	Partial Multiplier	Total, mm^2	relative to kP design, %	total mW	relative to kP design, %
130 nm	design1	0.28	5.40	basic	0.11	39.2	3.13	57.9
	design2	0.31	7.76	clas	0.14	45.8	5.49	70.8
	design3	0.28	5.60	ikm	0.11	39.8	3.33	59.5
	design4	0.29	6.01	combi1	0.13	43.1	3.74	62.3
	design5	0.29	6.1	combi2	0.13	43.0	3.79	62.5
250 nm	design1	1.38	40.9	basic	0.50	36.1	22.4	54.7
	design2	1.50	51.1	clas	0.61	41.0	32.6	63.7
	design3	1.40	42.7	ikm	0.51	36.7	24.2	56.5
	design4	1.46	45.1	combi1	0.58	39.5	26.6	58.8
	design5	1.46	46.0	combi2	0.57	39.4	27.4	59.6

Fig. 39 and Fig. 40 show the results of the analysis attacking traces of the ***kP*** designs with the 5 different partial multipliers as well as of the analysis attacking traces of just the corresponding field multipliers.

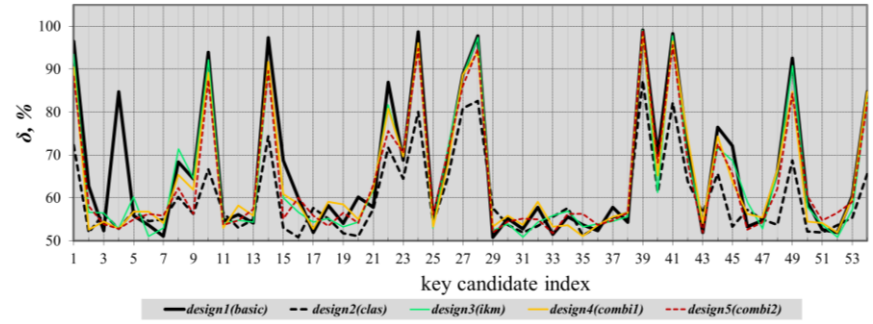


a) – attacking traces of the *kP* designs

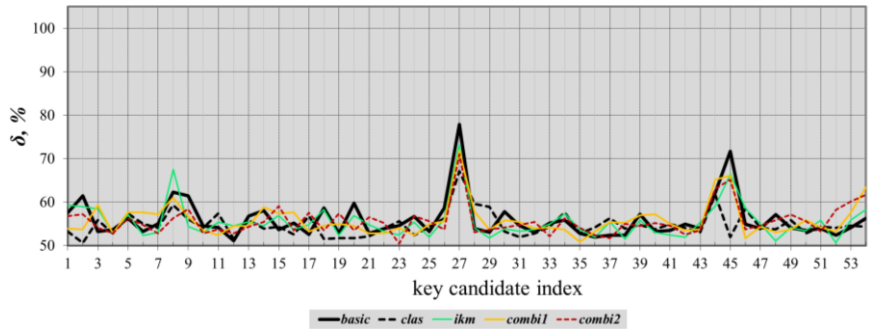


b) – attacking traces of the field multipliers

Fig. 39. Analysis results of traces of the *kP* designs with different partial multipliers synthesised for the IHP 130 nm technology: (a) – attacking traces of the whole *kP* design; (b) – attacking traces of the field multipliers.



a) – attacking traces of the *kP* designs



b) – attacking traces of the field multipliers

Fig. 40. Analysis results of traces of the *kP* designs with different partial multipliers synthesised for the IHP 250 nm technology: (a) – attacking traces of the whole *kP* design; (b) – attacking traces of the field multipliers.

The attack results for the IHP 130 nm technology clearly show that the ***kP*** design with the Partial Multiplier implemented using the classical multiplication formula is the best: no key candidates are extracted with a correctness higher than 60%, see dotted black line in Fig. 39. Designs with other Partial Multipliers have at least one key candidate with a correctness of 70%. The resistance of all implemented field multipliers is very high: there are no key candidates with a correctness higher than 60%, see Fig. 39-b.

The correctness of the revealed scalar for the IHP 130 nm technology using traces of the whole ***kP*** design is high in all investigated cases (see Fig. 40-*a*), but the ***kP*** design with the Partial Multiplier that was implemented using the classical MM is the most resistant one (see dotted black line). Results in Fig. 40-*b* show that the use of the classical MM for implementing the Partial Multiplier results in the most resistant field multiplier: the success rate of the attack is reduced from 78% correctness of the best key candidates down to 67%, see correctness of the $k_{candidate}^{27}$.

Thus, for both technologies, the ***kP*** design with the partial multiplier implemented using the classical multiplication formula is the best, i.e. the correctness of the extracted key candidates for this design (i.e. for the *design2*) is significantly smaller than for all other designs. For both technologies, the 5 implemented Partial Multipliers can be considered as resistant against our attack due to the fact that the correctness of all key candidates is smaller than 60% (this result is here not shown graphically).

Additionally, we calculated the Pearson correlation coefficients for the trace of the whole ***kP*** design and traces of the two components: ***BUS*** and ***MULT***. TABLE VII. shows the correlation coefficients for all 5 designs for both technologies.

TABLE VII. PEARSON COEFFICIENTS CALCULATED FOR THE WHOLE *KP* DESIGN AND ITS COMPONENTS BUS AND MULT

Technology	Trace of the component	trace of the ***kP*** design				
		design1	design2	design3	design4	design5
130 nm	*BUS*	0.80	0.65	0.79	0.80	0.79
	MULT	0.66	0.88	0.70	0.71	0.73
250 nm	*BUS*	0.75	0.64	0.74	0.74	0.72
	MULT	0.51	0.76	0.56	0.61	0.63

Only for the design with the Partial Multiplier implementing the classical MM the correlation coefficient for the whole ***kP*** design and its field multiplier is higher than the correlation coefficient of the ***kP*** design and its ***BUS*** (see the column marked with light green). Thus, in this design, the influence of the resistant component ***MULT*** is higher than the influence of the component ***BUS*** which is an SCA leakage source. So, the activity of the field multiplier hides the activity of the ***BUS*** at least partially.

6.1.2 **Design with randomized sequence of partial multiplications**

In this section we concentrate on the implementation details of the field multiplier, especially on the calculation sequence of partial products.

Usually, a field product of long operands is calculated as a sum of partial products of partial operands of smaller length. In each clock cycle one partial product is calculated and accumulated according to a fixed calculation sequence often denoted as an accumulation plan. In the IHP basic ***kP***-design the following fixed sequence for calculations of partial products *PP_i* was used corresponding to formula (33), i.e. corresponding to the 4-segment iterative Karatsuba MM proposed in 2004 in [77]:

$$
\begin{aligned}
&\text{clock cycle 1:} \quad PP\_1 = A_0 B_0 \\
&\text{clock cycle 2:} \quad PP\_2 = A_1 B_1 \\
&\text{clock cycle 3:} \quad PP\_3 = A_2 B_2 \\
&\text{clock cycle 4:} \quad PP\_4 = A_3 B_3 \\
&\text{clock cycle 5:} \quad PP\_5 = (A_0 \oplus A_1)(B_0 \oplus B_1) \\
&\text{clock cycle 6:} \quad PP\_6 = (A_0 \oplus A_2)(B_0 \oplus B_2) \\
&\text{clock cycle 7:} \quad PP\_7 = (A_1 \oplus A_3)(B_1 \oplus B_3) \\
&\text{clock cycle 8:} \quad PP\_8 = (A_2 \oplus A_3)(B_2 \oplus B_3) \\
&\text{clock cycle 9:} \quad PP\_9 = (A_0 \oplus A_1 \oplus A_2 \oplus A_3)(B_0 \oplus B_1 \oplus B_2 \oplus B_3)
\end{aligned}
\tag{34}
$$

In [60], [125] it was proposed to randomize the calculation sequence of the partial products, i.e. to re-schedule the calculation plan for each new field multiplication with the goal to increase the resistance of the field multiplier against DPA attacks. For the re-scheduling the enhanced multi-segment-Karatsuba (eMSK) multiplication formula with the segmentation of operands into 6 segments $eMSK_{6=2*3}$ as well as into 24 segments $eMSK_{24}$ was applied to calculate the field product of 192-bit long operands.

The $eMSK_{6=2*3}$ multiplication method is a combination of the original 2-segment Karatsuba MM [122] with the 3-segment Winograd MM [126]. The formula requires the calculation of 18 partial products of 32-bit long operands. The 18 partial multiplications can be performed sequentially in any order. If the area is the optimization parameter - only one partial multiplier for 32-bit long operands will be used. The calculation of the field product takes 18 clock cycles, and the designer defines the sequence in which the partial multiplications will be executed.

The $eMSK_{24}$ multiplication method requires the calculation of 162 partial products of 8-bit long operands. Thus, this MM requires at least 162 clock cycles but allows to use many more different sequences for calculating the partial products.

If the field multiplication formula consists of n partial products, there exist $n!$ different permutations of this sequence. The $eMSK_{6=2*3}$ multiplication method consists of 18 partial products, i.e. there exist 18! permutations of partial products, each of them results in a new calculation sequence. For the $eMSK_{24}$ multiplication method each of the 162! permutations of the calculations of the partial products can be used as a calculation sequence.

The field reduction has to be applied to the accumulation register and can be performed either once per field multiplication or after the calculation of each partial product. The latter design consumes more power for the calculation of the field product but the power shape of such a multiplication is more random. The partial reduction after each calculation of a partial product was implemented not only in [60], [125] but also in the designs reported in [77], [111] as well as in [66] so it can be considered as a usual way to do the implementation. Performing the reduction of the product after the accumulation of each partial product increases energy consumption that is a kind of a noise helping to reduce the success of SCA attacks.

The investigations [60], [125] showed that the field multiplier with the randomized sequence of partial products was more resistant against selected DPA attack but the resistance of a ***kP*** design with such a multiplier was not investigated. In [127] we investigated to which extent this kind of randomization increases the resistance of the whole ***kP*** design against selected horizontal attacks.

We experimented with the IHP basic ***kP*** design. Its field multiplier for 233-bit long operands was implemented corresponding to the 4-segment iterative Karatsuba MM [123]. Thus, the multiplication formula (33) was chosen for the re-scheduling.

Multiplication formula (33) consists of 9 different partial products listed in (34), i.e. there exists 9! = 362880 permutations of partial products, each of them results in a new calculation

sequence. One of these 9! possible permutations can be selected randomly for the calculation of each new field multiplication.

TABLE VIII. gives an overview of implementation details of our design and the implementation described in [60], [125].

TABLE VIII. OVERVIEW OF IMPLEMENTATION DETAILS OF TWO RANDOMIZED MULTIPLIERS

parameters	**design** [60], [125]	**our design**
field multiplier for:	$GF(2^r)$-elements	$GF(2^r)$-elements
lengths of multiplicands:	r=192	r=233
irreducible polynomial	not given	$f(t)=t^{233}+t^{74}+1$
number of segments	6	4
applied multiplication formula	$eMSK_{6=2*3}$	4-segment iterative Karatsuba MM
partial multiplier for:	32-bit long operands	59-bit long operands
number of partial multiplications	18	9
number of possible permutations	18!	9!

The smaller number of possible permutations in our design compared to the one described in [60], [125] means that our multiplier is more vulnerable to collision-based attacks. They are a kind of vertical attacks and can be prevented using traditional randomization countermeasures. In this work we concentrate on the prevention of horizontal DPA attacks. The area and energy consumption of a partial multiplier for 59-bit long operands are significantly higher than those of a multiplier for 32-bit long operands. Thus, the 59-bit partial multiplier can be more effective as a noise source and by that as a means against horizontal DPA attacks.

Additionally, we investigated the impact of the randomization of the field multiplier described here in combination with a partial multiplier implemented using the classical MM. For the evaluation we used the power traces simulated for designs synthesised for the IHP 250 nm gate library technologies. Thus, we analysed the power traces for the following 4 designs:

- *design1* that is the IHP basic ***kP*** design, see section 6.1.1.1;
- the *design1* with a randomized partial multiplication calculation sequence, further denoted as *design1_randomPartProductSequence*;
- *design2* that is the IHP basic ***kP*** design with partial multiplier implementing the classical MM, see section 6.1.1.2;
- the *design2* with a randomized partial multiplication calculation sequence, further denoted as *design2_randomPartProductSequence*.

We generated a random sequence for the partial product calculation for all field multiplications for the ***kP*** operation and save it as external data. We used these data for the randomization of the field multiplier in *design1_randomPartProductSequence* as well as in *design2_randomPartProductSequence*. The power traces for all 4 investigated designs were simulated using the same inputs, i.e. the key k and EC point P. Fig. 41 shows the results of the traces analysis applying the comparison to the mean method. The blue line represents the same results as in Fig. 40-*a* for the *design1(**basic**)*. The yellow line represents the same results as in Fig. 40-*a* for the *design2(**clas**)*.

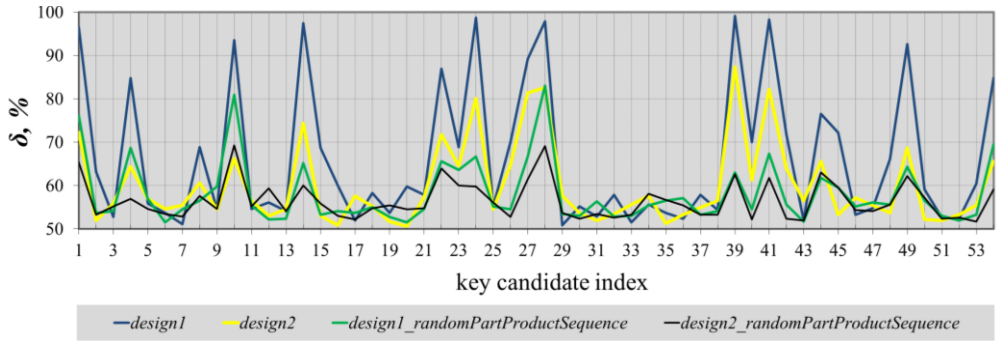


Fig. 41. Analysis results.

In Fig. 42 the key candidates are sorted in descending order of their correctness. According to that, each key candidate got a new index displayed at the *x*-axis. This representation helps to compare the analysis results.

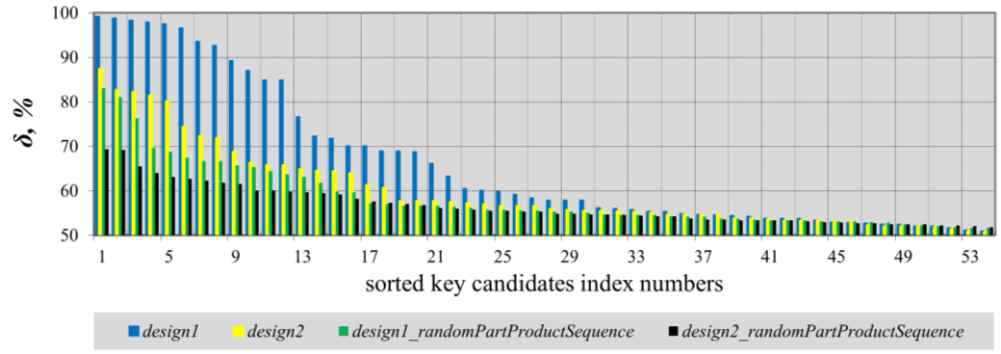


Fig. 42. Attack results: correctness of the extracted keys sorted in descending order.

The results of the analysis show that the implementation of the partial multiplier using the classical multiplication formula significantly reduces the success of the horizontal attacks (compare yellow and blue bars in Fig. 42). This effect is similar to the randomization of the calculation sequence of partial products (compare yellow and green bars). Both strategies combined, i.e. applying a randomized sequence of partial multiplications and implementing the partial multiplier using the classical MM increase this effect significantly: the correctness of the extraction was decreased from 99% for the IHP basic $k\boldsymbol{P}$ design (see blue bars for *design1*) to 69% (see black bars) that is a significant improvement of the design's resistance against the applied horizontal SCA attack.

Please note that the randomization of calculation sequence of the partial products is easy to implement only for the multiplication of the $GF(2^n)$-elements [128], which is due to the fact that for $GF(2^n)$ the accumulation of the partial products is a bitwise XOR operation b. This randomization strategy can be applied also for the multiplication of *GF(p)*-elements, but the sign of each partial product (positive or negative) has to be taken to account by accumulation. This can reduce the number of permutations significantly.

6.1.3 Discussion of the influence of the Multiplier on the design's resistance

The results of the analysis presented in section 6.1.1.3 demonstrate clearly that the implemented formula for partial multiplication has a significant impact on the resistance of the $k\boldsymbol{P}$ design against horizontal attacks. The design with its partial multiplier implemented *using* the classical multiplication method (i.e. *design 2*) is the "most resistant", i.e. applying the classical

MM for implementation of the partial multiplier is a kind of countermeasure against horizontal attacks. The randomization of the sequence calculating partial products increases the resistance of the whole ***kP*** design additionally. Combining these two approaches allows to increase the resistance of the ***kP*** design significantly. This is due to the fact that compared to the other designs investigated, not only the average power consumption of designs with the classical partial multiplier, but also the amplitude range of the power trace, is the highest one.

Fig. 43 shows power traces for the 4 designs investigated in the previous section. The *x*-axis shows the execution time, in clock cycles. Grey lines represent the power traces of the ***kP*** execution for the whole design. Orange lines correspond to the power traces of *the* field multiplier and the blue lines to the power traces of the partial multiplier. Please note that here the compressed power traces are shown: each clock cycle is represented with only one value that is a sum of all 300 sample power values within the clock cycle (i.e. the *y*-axis shows the average power consumption in W, multiplied by 300).

It is clear that the ***basic*** partial multiplier (see blue lines in Fig. 43-*a* and Fig. 43-*c*) consumes less energy than the ***clas*** partial multiplier (see blue lines in Fig. 43-*b* and Fig. 43-*d*). Moreover, the amplitude range of the power trace of the ***clas*** partial multiplier is about 2-3 times higher than the one for the ***basic*** partial multiplier (compare blue lines in Fig. 43-*a* and Fig. 43-*b*). These differences cause similar differences in the power consumption of the field multipliers: the field multiplier of *design2* consumes more power than the one of the *design1*, whereby the amplitude range of its power trace is also significantly higher (compare orange lines in Fig. 43-*a* and -*b*). Consequently, the amplitude range of the power trace of the whole *design2* is also higher than the one of the *design1* (compare grey lines in Fig. 43-*a* and -*b*). The influence of the randomization on the power traces of the partial multipliers is low: no visible differences in blue lines in Fig. 43-*a* and -*c*; similarly, no visible differences in blue lines in Fig. 43-*b* and -*d*. The randomization influences the power traces of the field multipliers significantly (compare orange lines in Fig. 43-*a* and -*c*) and – consequently – the power traces of the whole designs (compare grey lines in Fig. 43-*a* and -*c*).

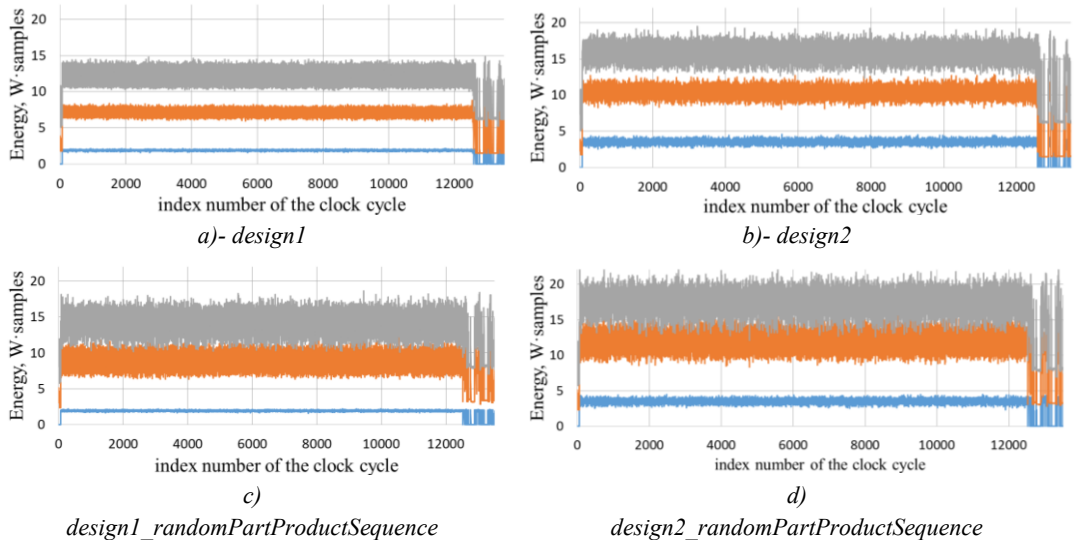
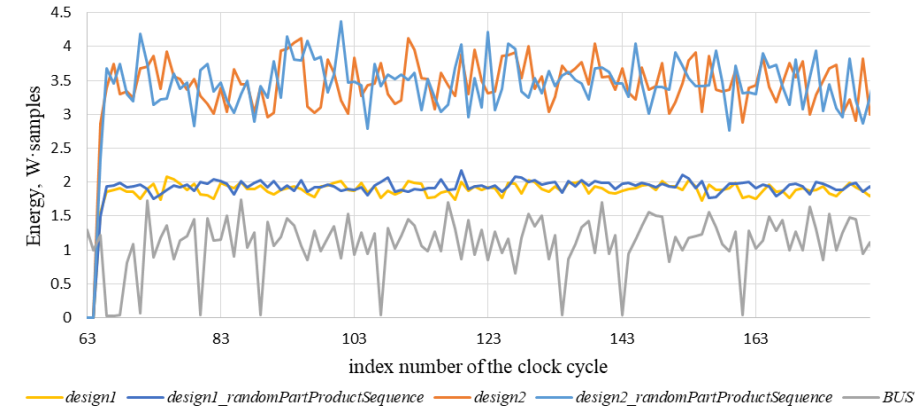
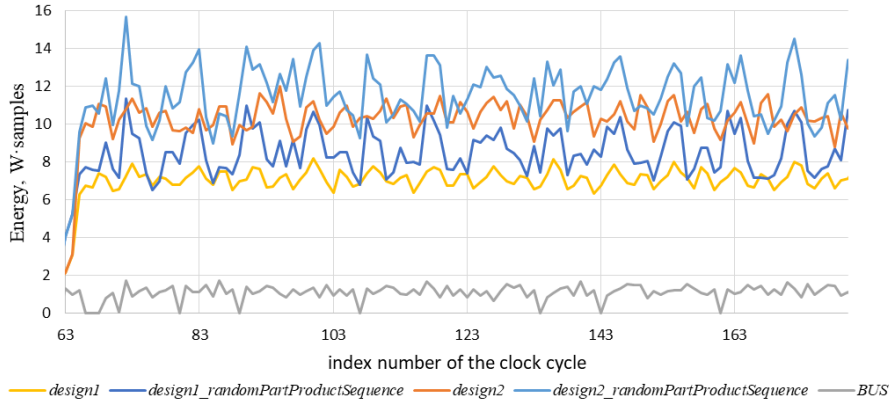


Fig. 43. Power traces simulated for different designs and their blocks: grey line is the PT of the whole ***kP*** design; orange line is the PT of the field multiplier; blue line is the PT of the partial multiplier.

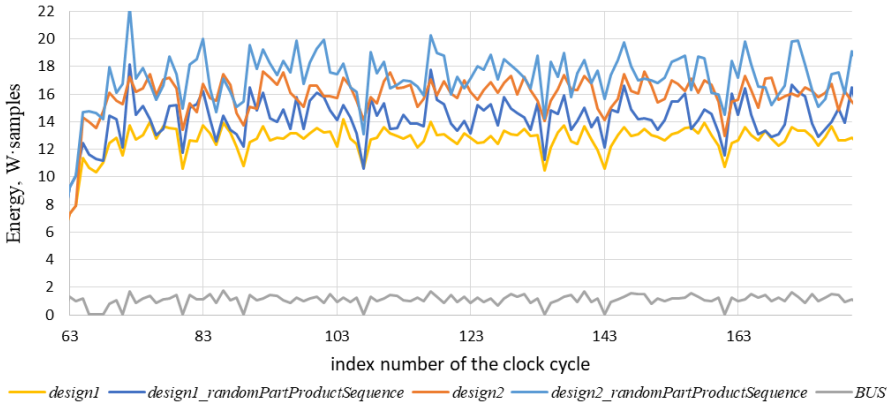
Fig. 44 shows a part of investigated power traces, for the whole designs and selected blocks. The grey line is the power trace of the block BUS that is the strongest SCA leakage source.



a) – a part of power traces of partial multipliers, zoomed in



b) – a part of power traces of field multipliers, zoomed in



*c) – a part of power traces of whole k**P** designs, zoomed in*

Fig. 44. Power traces simulated for different designs and their blocks, zoomed in.

It can be clearly seen that the power consumption of the BUS (see grey line in Fig. 44-*a*) is almost as high as the one of the ***basic*** partial multiplier (see yellow line in Fig. 44-*a*). The peak-to-peak amplitude variation range of the BUS's power trace is changing considerably over time. Therefore, it influences significantly the shape of the power trace of the whole *design1* (compare the shapes of grey and yellow lines in Fig. 44-*c*) causing the success of the horizontal attacks.

Due to the increased amplitude range of the ***clas*** partial multiplier and – consequently – of the *design2,* the activity of the BUS influences the shape of the *design2* less than the one of the *design1*. Thus, the classical partial multiplier "hides" the activity of the BUS, at least partially. Similarly, the increased amplitude range of the power traces of the randomized field multipliers hides the activity of the BUS, i.e. such designs are inherently more resistant against horizontal attacks. Please note, that the hiding effect of the randomized field multiplier can be not so high if the field reduction will be executed only once per field product calculation, i.e. after the accumulation of all partial products. But we did not investigate this effect in this work.

6.2 **Regular scheduling**

As it was shown in the previous section, the field multiplier can be used as a kind of noise that hides the activity of the other design blocks and – consequently – reduces the success of the horizontal attacks. We assume that the resistance against vertical SCA attacks also will be increased, due to the hiding effect of the multiplier[21], but the application of the multiplier as a single countermeasure is not effective enough to avoid successful SCA attacks. Thus, a re-design of the block(s) that are strong SCA leakage sources is unavoidable.

In section 2.4.3 the blocks that caused the success of the horizontal attacks are listed. The ***BUS*** is a big block of the investigated design. Its power consumption is high and comparable to the one of the partial multiplier, i.e. it influences the shape of the power trace of the whole ***kP*** design significantly (see section 6.1.3). Due to these facts, the ***BUS*** was selected as the first candidate for the re-design (see section 4).

In this section we concentrate on the block ***BUS***, and describe at first the implementation details with which we experimented to reduce its vulnerability to horizontal attacks.

In the IHP basic ***kP*** design (i.e. in the *design1*) 12 different addresses are reserved for the following blocks: 10 registers, ***ALU*** and ***MULT***. TABLE IX. contains the address of each block of the *design1*, as hexadecimal numbers.

TABLE IX. ADDRESSES OF BLOCKS IN THE IHP BASIC *kP* DESIGN

block	***MULT***	***ALU***	***REGISTERs***									
			x	***y***	***k***	***b***	$\mathbf{X_1}$	$\mathbf{X_2}$	$\mathbf{X_3}$	$\mathbf{X_4}$	$\mathbf{Z_1}$	$\mathbf{Z_2}$
address	6	7	8	9	0	1	4	5	A	B	2	3
functionality of the block	field multiplication	field addition and squaring	input: *x*-coord. of point ***P*** (const during a ***kP*** execution) output: *x*-coord. of point ***kP***	input: *y*-coord. of point ***P*** (const during a ***kP*** execution) output: *y*-coord. of point ***kP***	processed scalar (key)	A public parameter of EC *B-233*, const	Registers used in the main loop for the storing of intermediate data					

Fig. 45 shows the processing sequence in the main loop of the IHP basic ***kP*** design providing details about the activity of each block. The rectangles represent different activities in the design. Rectangles that are vertically aligned are processed in parallel, i.e. in the same clock cycle. All *write-to-register* operations are depicted by small green squares for the addressing of the register in one clock cycle and a small grey square for the storing operation in the next

[21] We did not investigate this effect in this work.

clock cycle. Red rectangles show the activity of the field multiplier. In our implementation the multiplier is always active, i.e. obtaining both multiplicands is performed in parallel to the calculation of the last two of the 9 partial products. The activity of the block ***ALU*** is shown using yellow rectangles for the squaring operation and blue rectangles for the field addition.

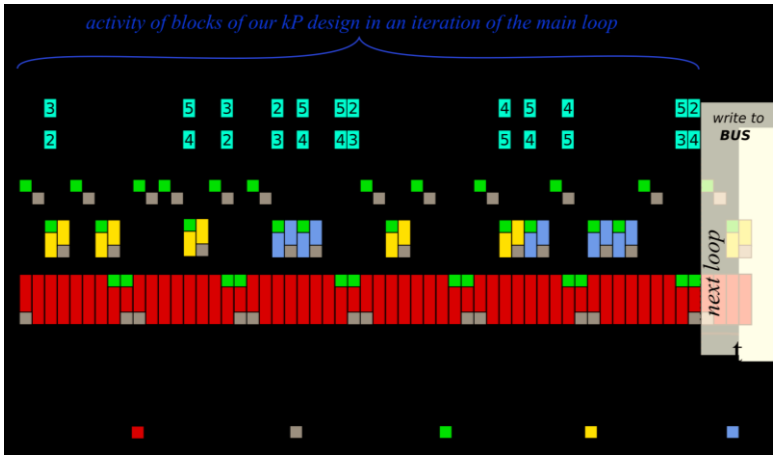


Fig. 45. Processing sequence of the main loop in the IHP basic *k**P*** design: details about the activity of each block and the key-dependent addressing of the blocks for *write-to-bus* operation.

A detailed description of the main loop implemented in *design1* is given in Appendix 1. For the investigation described in this section the activity of the ***BUS*** is important. The ***BUS*** is realized as a muxer. It consists of many logic gates that react on the address given by the ***Controller***. The *write-to-bus* operation connects the output of the addressed block to the inputs of all other blocks. By the *read-from-bus* operation, only the addressed block accepts the values on its input as data for processing.

The numbers in rectangles in Fig. 45 represent the addresses of the blocks of the *design1*. The sequences of these numbers differ for $k_i=0$ and $k_i=1$ and show which block writes to the ***BUS*** in the corresponding clock cycle. Due to the fact that the use of registers in the Montgomery *k**P*** algorithm depends on the processed bit value k_i of the scalar k, horizontal SCA attacks can be successful. This dependability is shown by the turquoise-marked numbered rectangles in Fig. 45.

Please note that in Fig. 45 not all clock cycles are marked as potentially dangerous with respect to the addressing of the blocks. For example, there are unmarked clock cycles that lead to high correctness of the extracted key, i.e. 8 and 9. The reason of the high attack success in these clock cycles is also the key-dependent addressing of the blocks of the design. But this dependability is more complex, due to the fact, that the IHP basic *k**P*** design has 4 variants of main loop operation sequences. The selection of one of the four sequences depends on the current key bit value to be processed as well as on the previous key bit value. Appendix 1 shows this dependability in detail. A high level of details of a description of an implementation important for the designers, who cope with those issues but it makes the explanation voluminous and complex. Thus, we decided to describe the problem and our proposed solution without a comprehensive explanation of all implementation details, instead we selected typical SCA leakage sources and marked the corresponding clock cycles as potentially dangerous (turquoise) in Fig. 45.

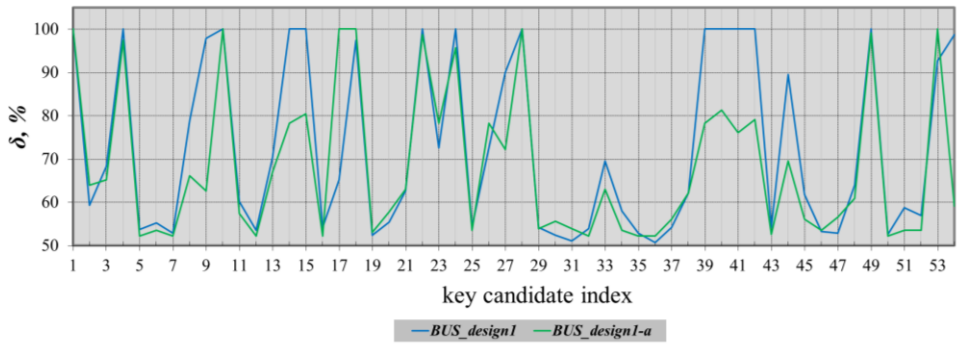
For example, in clock cycle 3 of the main loop, register Z_2 has to write its content to the ***BUS*** if $k_i=1$ (see the first rectangle marked with number 3 in Fig. 45). If $k_i=0$ register Z_1 (its address is 2) is selected to write its content to the ***BUS***. In most cases, i.e. in 42 out of 54 clock cycles, the addressing of the blocks doesn't depend on k_i.

There exist clock cycles in our design in which none of the blocks reads from or writes to the ***BUS***, see for example the 2nd and the 4th clock cycle. In these clock cycles any or none of the blocks can write to the ***BUS*** without influencing the functionality of the ***kP*** implementation. In our implementation the block with address 0, i.e. the scalar *k* is always written to the ***BUS***.

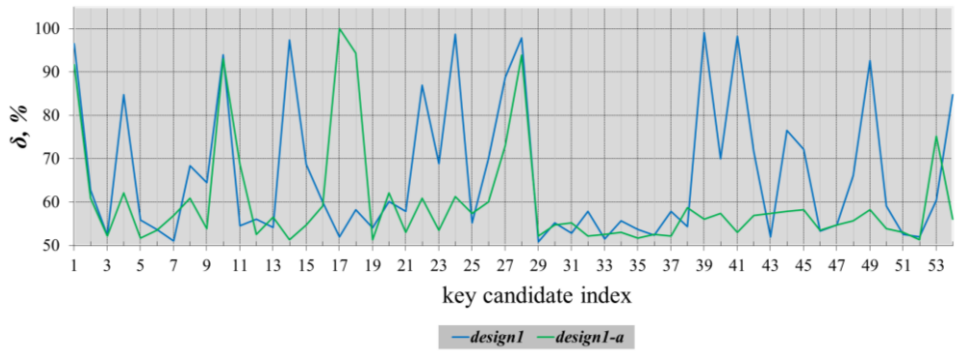
We assume that this behaviour is the reason why the horizontal SCA attacks against the IHP basic ***kP*** design are successful. So, our hypothesis for coping with that issue is that if another block will be addressed and its output value is not a constant one the success rate of the attacks can be reduced. If different blocks will be addressed the success rate of the attacks will be reduced further.

In this section we describe our experiments with four ***kP*** designs. The designs differ only in the way of addressing and/or addressing sequence of blocks for the *write-to-bus* operation.

At first, we experimented with the representation of the addresses for the *write-to-bus* operation in the block Controller. In the IHP basic ***kP*** design only 4 bits are reserved in the Controller for the block addressing [79]. The hamming distance when changing the address from 3 to 0 (see for example 3rd and the 4th clock cycles in case $k_i=1$) is 2, while in case $k_i=0$ the hamming distance is 1, due to changing the address from 2 to 0. To evaluate the influence of the hamming distance on the attack success, we extended the length of the addresses in the Controller: we reserved 12 bits for addressing of the 12 design blocks, in order to make the hamming distance by changing the addresses always equal to 2. Due to the fact that this change in the design was not significant, we denoted this modified *design1* as *design1-a*. We synthesized the *design1-a* for the IHP 250 nm technology using exactly the same parameters as for the synthesis of *design1*. Fig. 46 shows the analysis results for both designs.



a)- attack results of the ***BUS*** in *design1* (blue line) and in *design1-a* (green line)



b) – attack results of whole ***kP*** designs

Fig. 46. Analysis results attacking simulated power traces for *design1* (blue lines) and *design1-a* (green lines): *(a)* – results of the attack of power traces of the block ***BUS*** in both designs ; *(b)* – results of the attacks of power traces of the whole *design1* and *design1-a* .

The attack success analysing only the BUS's power traces is reduced. Analysis results of the power traces of the whole ***kP*** designs are not so manifestly: in many clock cycles the attack success is reduced (see for example 4, 22, 23, 24, 27, 39, and so on), but in clock cycles 1, 10, 17, 18, 28 and 53 the attack was more successful for the modified IHP ***kP*** design, i.e. for the *design1-a.* In each case, the attack results demonstrate that even insignificant changes in the way of addressing the blocks of the design influence the resistance of the whole design significantly. Due to this fact, and with the goal to verify the idea about the nature of the success of the horizontal attacks we experimented further with the implementation of addressing of the blocks and implemented two additional designs: *design_63* and *design1_regular*.

Fig. 47 shows the addressing sequence for all designs investigated here. The addressing sequence for the *design1* is again shown in Fig. 47-*a*. In our *design_63* the block with address 0 no longer writes to the bus. Instead either the block with address 6 or the block with address 3 writes to the bus (see Fig. 47-*b*). The addressing of all other blocks is the same as in *design1*. In our *design1_regular* within the main loop iteration (i.e. for each slot) we implemented an addressing sequence in which instead of the block with address 0 the blocks with addresses from 1 up to 11 (1 to B in hexadecimal) are selected always in a fixed sequence, i.e. one after the other. In the *design1* there are 21 clock cycles in which address 0 is active. This means that in the first main loop iteration the sequence of the block addresses starts with 1 and ends with 10 (hexadecimal A). In the second main loop iteration we continued the sequence starting with 11 (hexadecimal B) ending with 9. In the third main loop iteration it starts with 10 (hexadecimal A) and ends with 8, and so on. Thus, in *design1_regular* all blocks are addressed equally often within a ***kP*** execution. The addressing sequences for the first two key bits processed in the main loop (i.e. for different values k_{n-3} and k_{n-4}) are completely shown in Fig. 47-*c*. For the key bits k_{n-5}, k_{n-6}, and k_{n-7} the beginning of the main loop iteration is shown to illustrate the implemented regular addressing sequence.

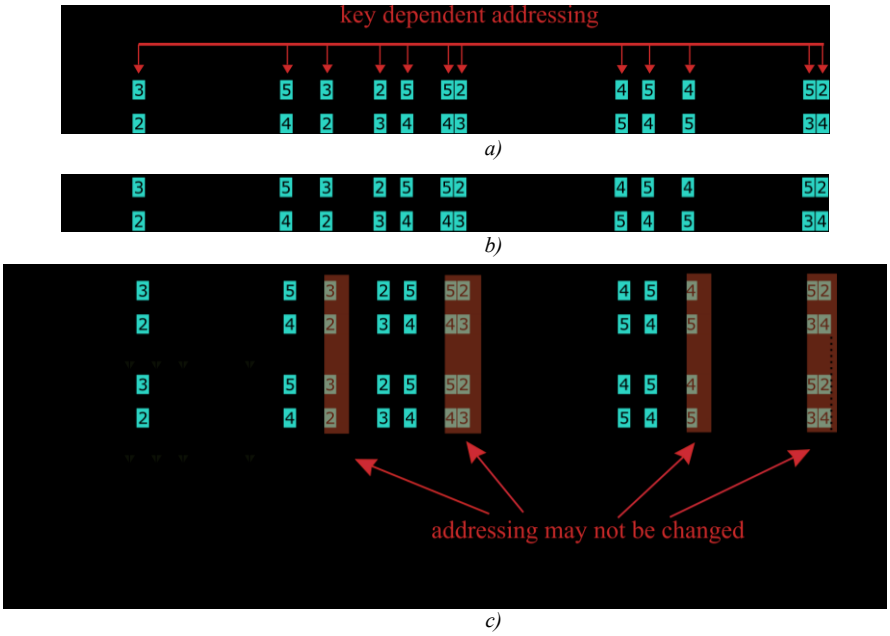


Fig. 47. Details about the main loop implementation in our ***kP*** designs: the sequences of the numbers show for each implemented design which block writes to the ***BUS*** in the corresponding clock cycle. Sequences for *design1*, *design_63* and *design1_regular* are shown in *(a)*, *(b)* and *(c)* respectively.

For the *design_63* we analysed the PT simulated for the IHP 250 nm technology. Fig. 48 shows a comparison of the analysis results for the *design1* (see blue line) and *design_63* (see

red line). Here it can be clearly seen that the changes made in the addressing of blocks influence the resistance of the ***kP*** design in many clock cycles significantly but in total it does not decrease the success of the attack.

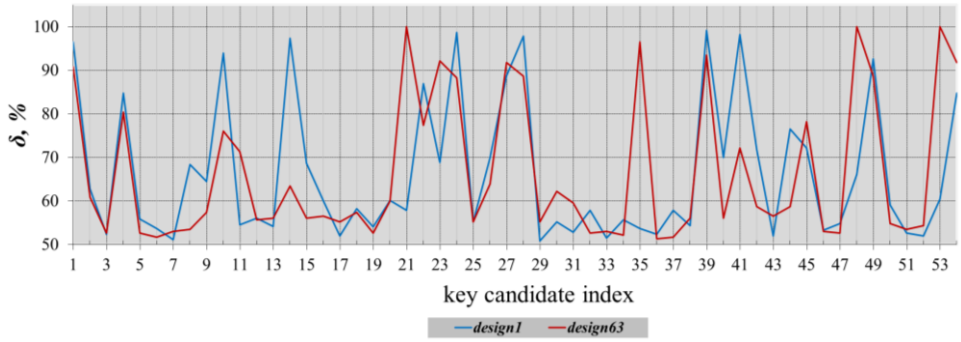
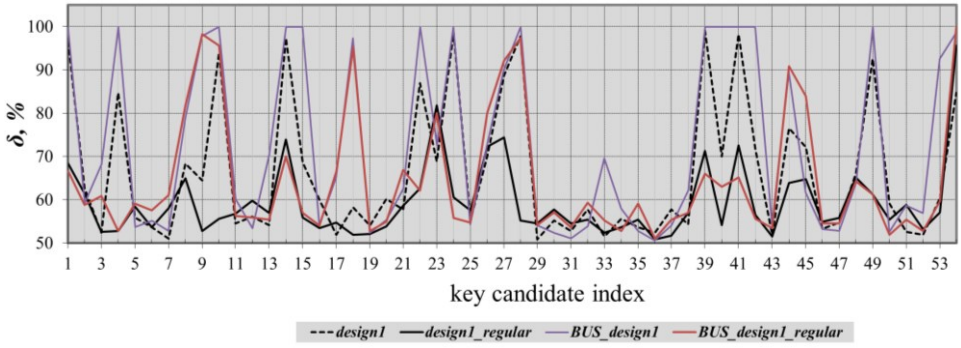
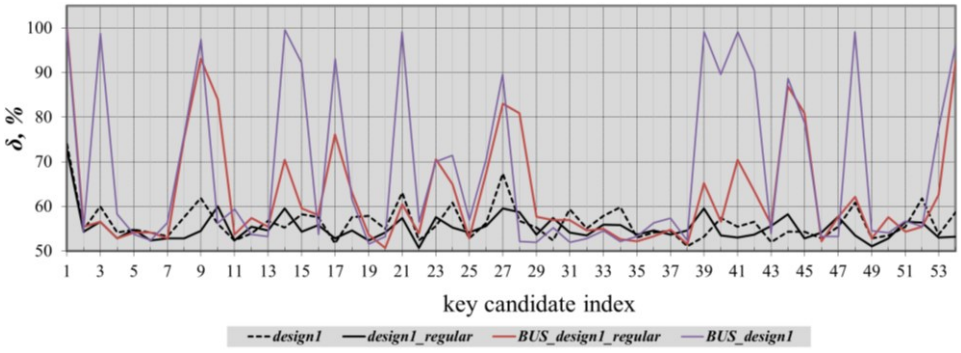


Fig. 48. Analysis results attacking simulated power traces for *design1* (blue line) and *design_63* (red line): whole ***kP*** designs are attacked (simulation for the IHP 250 nm technology).

We synthesized the design applying the regular schedule, further denoted as *design1_regular,* for both IHP technologies. Fig. 49 shows the results of the analysis for the whole ***kP*** design (see solid black line) and for the ***BUS*** (see red line), in comparison to the *design1*.



a) 250 nm technology



b) 130 nm technology

Fig. 49. Analysis results attacking *design1* and *design1_regular: a) 250nm technology, b) 130nm technology*.

Fig. 49 clearly shows that the success of the performed horizontal attack for the 250 nm technology is significantly reduced for the design with the regular schedule of addresses compared to our basic design (see the black dotted and black solid lines in Fig. 49-*a*). Also, for the design in the 130 nm technology, the success rate of the attack analysing the trace of the ***BUS*** is significantly reduced (see the red and violet lines in Fig. 49-*b*).

Comparing the Pearson coefficients calculated for ***kP*** and ***BUS*** only traces shows that the regular schedule reduces the influence of the ***BUS*** on the shape of the ***kP*** trace, whereby the influence of the ***MULT*** is slightly increased for both technologies, see TABLE X. The impact of the change in the influences is manifested in the reduced success of the horizontal attack performed.

Please note that in our designs due to the implemented logic not the complete sequence of the addressing of components can be adapted. This holds true for the following sequence of clock cycles: 7-10, 16-19, 26-28, 35-37, 43-46, 53-54-1. The 5 key candidates with the highest correctness obtained using the PT of the *design1_regular* are 10, 17, 18, 28 and 45, i.e. they are part of the clock cycle sequences that cannot be modified. Due to this fact, a significant reduction of the success rate of the attacks for these clock cycles was not expected.

TABLE X. ANALYSIS RESULTS FOR THE DESIGN WITH THE REGULAR SCHEDULE

design		Number of key candidates extracted with a correctness δ≥90%		Pearson coefficient	
		kP design	***BUS***	***kP to BUS***	***kP to MULT***
250 nm	basic design	7	19	0.75	0.51
	design with the regular schedule	1	7	0.69	0.57
130 nm	basic design	0	18	0.80	0.66
	design with the regular schedule	0	3	0.66	0.68

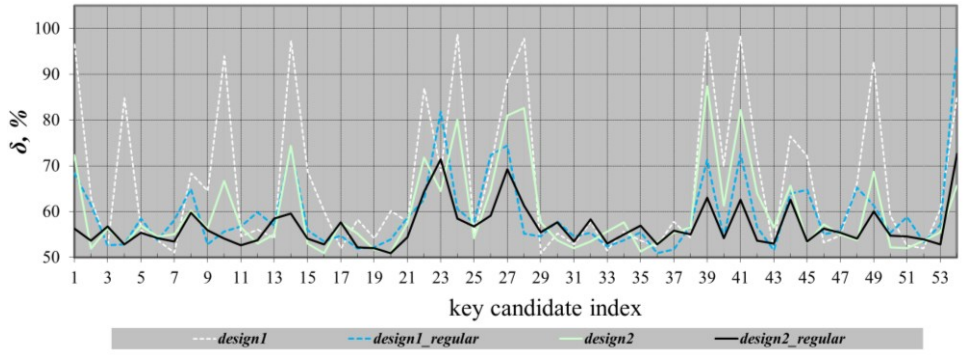
In this section we investigated how some implementation details of the *BUS* influence the success of horizontal attacks, for deriving a possible re-design strategy. The results of our experiments demonstrate that the hamming distance as well as the length reserved for the addresses of the design block influence the attack success, but this influence does not reduce the success of the attack in total that clearly: SCA leakage was significantly reduced in some clock cycles but increased in other clock cycles. Thus, this effect is not a universal means for all clock cycles but this strategy can be successfully used in combination with other approaches.

The proposed regular scheduling [129]-[130] for the block addressing is an effective strategy reducing the success of the horizontal attacks in all clock cycles where it is applicable.

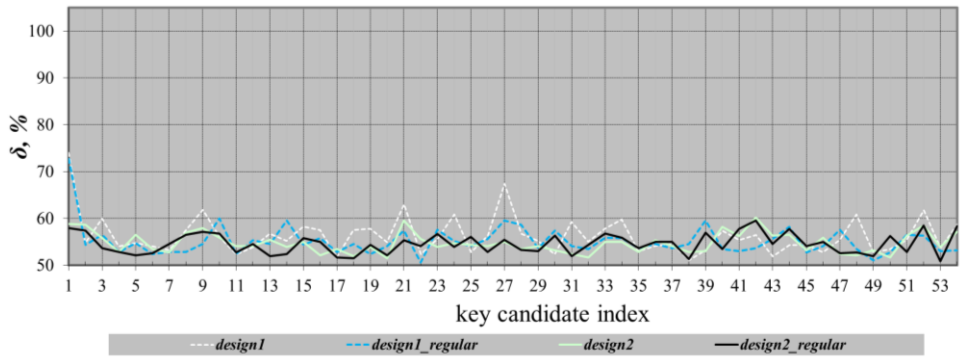
6.3 Combining Countermeasures

In this section we describe the results of the analysis attacking our ***kP*** design in that we combined the regular scheduling for the block addressing as described in section 6.2 with the most resistant field multiplier that contains the partial multiplier implemented using the classical multiplication formula (see section 6.1.1). We denote this design *design2_regular*.

Fig. 50 shows the results of the analysis for *design2_regular* in comparison to the other designs: the IHP basic ***kP*** design (*design1)*, the design with the classical partial multiplier *design2*, and the design with regular scheduling of the block addressing *design1_regular*. The white dotted line corresponds to *design1*; the blue line to *design1_regular*, the green line to *design2* with the "most resistant" multiplier. The solid black line shows the success of the attack for our last design, i.e. for the design combining the regular schedule with the "most resistant" multiplier, i.e, for *design2_regular*.



a) 250 nm technology



b) 130 nm technology

Fig. 50. Analysis results of attacking the designs: *design1, design2, design1_regular* and *design2_regular*.

Fig. 50-*a* clearly shows that the success of the performed horizontal attack for the 250 nm technology is significantly reduced for our last design, see the black line. Only 3 key candidates with a correctness of 71%, 69% and 73% are extracted, see $k_{candidate}^{23}$, $k_{candidate}^{27}$, $k_{candidate}^{54}$. The correctness of other key candidates is between 50% and 63%. For the design in 130 nm technology, the success rate of the attack is really small: the correctness of all key candidates is between 50% and 60%, see black line in Fig. 50-*b*. TABLE XI. shows the number of the key candidates extracted with a correctness of more than 70% for the ***kP*** design in both technologies and Pearson coefficients between the whole ***kP*** design and its ***BUS*** and its ***MULT*** respectively.

TABLE XI. ANALYSIS RESULTS FOR THE *KP* DESIGN WITH THE REGULAR SCHEDULE AND THE "MOST RESISTANT" MULTIPLIER (*DESIGN2_REGULAR*)

Technology	Number of key candidates extracted with a correctness δ≥70%	Pearson coefficient	
		kP to BUS	***kP to MULT***
250 nm	2	0.53	0.80
130 nm	0	0.48	0.90

The comparison of the correlation coefficients for all designs investigated here shows that in our last design the influence of the ***BUS*** is the smallest one and the influence of the multiplier is the highest one for both IHP technologies. These results we published in [131].

6.4 Influence of the design synthesis options

Due to the fact, that different circuits implementing the same functionality result in different power consumptions, the different design synthesis options can influence the resistance of the cryptographic circuits significantly. We have chosen two of the compilation techniques in the Synopsys Design Compiler (version K-2015.06-SP2) [82] that are available to perform

hardware optimizations: simple *compile* option and *compile_ultra*. The *compile* optimization option aims at performing an area optimisation whereas the *compile_ultra* option has to be applied for the synthesis of area- and power-optimised designs. In this section we investigated how an application of these options influences the resistance of the ***kP*** design. Please note that there is no detailed explanation available on how exactly different Design Compiler options work, i.e. how they affect the circuitry. Thus, the reasons of the circuit changes cannot be investigated and explained in detail. Nevertheless, designers have to be aware that the application of different synthesis techniques and options influences the resistance of cryptographic circuits. This section demonstrates this fact. The results are also described in [132]-[133].

We synthesized the IHP basic ***kP*** design using the gate library for the IHP 250 nm technology for a clock frequency of 20 MHz (i.e. for a clock cycle duration of 50 ns). Please note that the designs described in previous sections were synthesised for a clock cycle of 30ns applying the simple *compile* option. In the rest of this work, we denote the IHP basic ***kP*** design synthesized for the clock frequency of 20 MHz applying the simple *compile* option as *D_noUltra*. Additionally, we synthesised the same design for the same clock frequency of 20 MHz, using the *compile_ultra* option and denoted here as *D_ultra*.

The *compile* command performs logic-level and gate-level synthesis and optimization of the current design. The optimization process trades off timing and area constraints to provide the smallest possible circuit that meets the specified timing requirements. Values for area and speed of components used while synthesizing and optimizing the design are obtained from user-specified libraries.

The *compile_ultra* command performs a high-effort compilation on the design to achieve an optimum quality of results by enabling all DC Ultra features, therefore increasing the performance and significantly reducing design area and power consumption. It is targeted towards high-performance designs with very tight timing constraints.

We simulated the power consumption of the two ***kP*** designs after syntheses and after layout. Both designs investigated here, i.e. *D_noUltra* and *D_ultra*, process the same input data, i.e. the scalar k and the EC point $\boldsymbol{P}$ (see values in section 3.1.1), and require the same amount of clock cycles (about 13000) for a single ***kP*** operation. All ***kP*** traces were simulated using the Synopsys PrimeTime suite.

TABLE XII. presents the parameters of the investigated ***kP*** designs. Despite the similar area, the power consumption of the investigated ***kP*** designs obtained after the synthesis and after the layout are quite different.

TABLE XII. PARAMETERS OF INVESTIGATED *KP* DESIGNS

	Investigated *kP* designs			**Field multiplier**			
				area		***power***	
	name	***area, mm²***	***power, mW***	***total, mm²***	***relative to kP design***	***total, mW***	***relative to kP design***
Synthesis	*D_noUltra*	1.78	21.1	0.69	37.4 %	14.6	69.2 %
	D_ultra	1.68	13.7	0.62	36.8 %	8.58	62.5 %
Layout	*D_noUltra*	1.77	25.5	0.67	37.6 %	16	62.7 %
	D_ultra	1.70	16.6	0.63	37.0 %	8.5	51.1 %

We compressed the simulated power trace, i.e. we represented each clock cycle using only one value – the average power value of the clock cycle. Fig. 51 and Fig. 52 show compressed simulated traces obtained after synthesis and after layout, respectively. Power traces for *D_noUltra* are shown in red and power traces for *D_ultra* in blue. It can be seen that the design obtained using the *compile_ultra* technique consumes significantly less power.

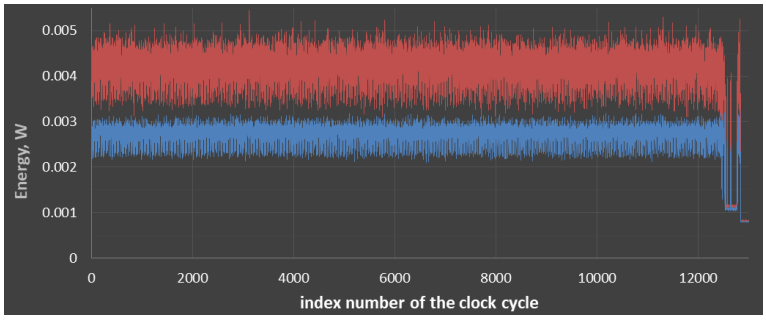


Fig. 51. Compressed simulated power traces for the design *D_noUltra* (red line) and for the design *D_ultra* (blue line): the traces were simulated after synthesis.

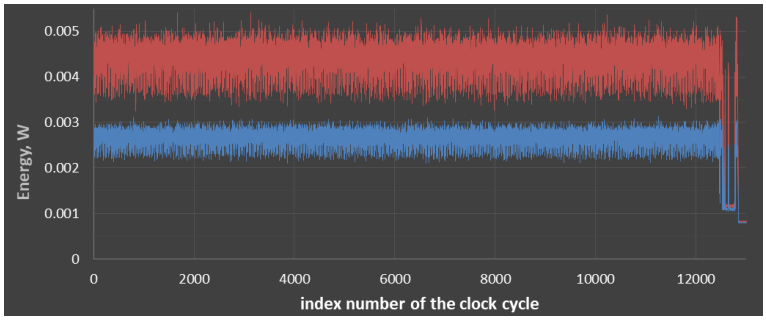


Fig. 52. Compressed simulated power traces for the design *D_noUltra* (red line) and for the design *D_ultra* (blue line): the traces were simulated after layout.

All 4 compressed traces were analysed using *the comparison to the mean* method. Fig. 53 shows the calculated correctness of the key candidates obtained by analyzing traces simulated after synthesis, i.e. traces shown in Fig. 51 were analysed. Fig. 54 shows the attack results analyzing traces simulated after layout, i.e. traces shown in Fig. 52 were analysed.

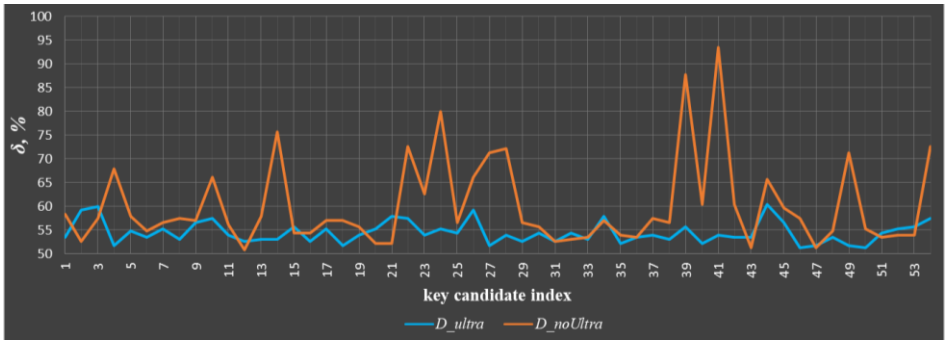


Fig. 53. Horizontal attack using *the comparison to the mean* method: the analysis results are obtained using traces after synthesis of the ***kP*** designs *D_noUltra* (orange) and *D_ultra* (blue) synthesized for the 250 nm technology.

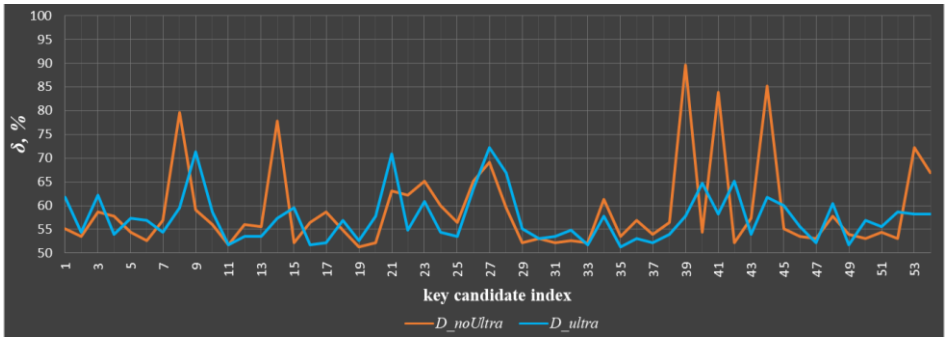


Fig. 54. Horizontal attack using *the comparison to the mean* method: the analysis results are obtained using traces after layout of the ***kP*** designs *D_noUltra* (orange) and *D_ultra* (blue) synthesized for the 250 nm technology.

For the IHP basic ***kP*** design synthesised with the *complile* option (i.e. for the design *D_noUltra*) the correctness of the best key candidates is 93% and 90% for the analysis of traces simulated after the synthesis and after the layout respectively. Please note that the analysis results for the same design attacking both traces differ significantly too, i.e. different clock cycles are the strong SCA leakage sources. For example, the correctness of the key candidate 44 after synthesis is about 65% only, but after layout it is about 85% (compare the orange lines in Fig. 53 and Fig. 54). A similar effect, i.e. a significantly increased attack success when analysing the trace after layout, is observable also for key candidate 8. But there are some key candidates, for example those with the index numbers 22, 24, 49, for which the correctness of the revealed key candidates was between 70% and 80% when attacking the trace after synthesis but for which the correctness dropped down to about 60% when attacking the trace after layout.

For the design *D_ultra* the attack results analysing the trace simulated after synthesis show a high resistance of the design against the performed attack: the correctness of the best key candidates is about 60%. Attacking the trace simulated after layout we obtained a few key candidates with increased correctness, in comparison to the trace simulated after synthesis. But the correctness of the best 3 key candidates those with index numbers 9, 21 and 27 is about 72 % only. So, the optimization performed using the *compile_ultra* option hides information leakage significantly, at least for the design investigated here.

The designs described here were also manufactured in the IHP 250 nm technology. Additionally, we ported both designs on an FPGA. We analysed the power and electromagnetic traces measured on the ASICs as well on the FPGA. For the analysis we used not only compressed but also uncompressed traces and applied more statistical methods. Additionally, we investigated the influence of the clock frequency on the resistance of the designs. The experiments with the manufactured ASICs and with the FPGA designs are described in section 7.

7 Experiments with own ASICs and FPGAs

In section 6 we described our ideas for increasing the resistance of the investigated ECC designs and evaluated the ideas using only the simulated power traces. In this section we evaluate our ideas practically, i.e. analysing the traces measured using the ASICs manufactured in the IHP 250nm technology as well as FPGA implementations.

7.1 ASICs

With the goal to evaluate the efficiency of the proposed countermeasures (see section 6) practically, i.e. analysing measured traces, some designs were selected for manufacturing in the IHP 250 nm technology. The manufactured ASICs are:

- ASIC1 is a ***kP*** design with the classical partial multiplier as described in 6.1.1.1 (see *design2* in 6.1.1.1), whereby the usual *compile* option was applied for the design synthesis;
- ASIC2 is a ***kP*** design with the classical partial multiplier as described in 6.1.1.1 synthesized using the *compile_ultra* option, i.e. the difference between ASIC1 and ASIC2 is only this compiler option applied for the design synthesis (see section 6.4);
- ASIC3 is the ***kP*** *design2_regular*, with a classical partial multiplier as described in 6.1.1.1 and the regular addressing schedule as described in 6.1.3. The influence of the combination of these two countermeasures on the resistance to horizontal attacks was discussed in section 6.3. The design was synthesised with the same compiler options as for the synthesis of the ASIC1. Thus, the difference between the ASIC3 and ASIC1 is the addressing of the design blocks for the write-to-bus operation in selected clock cycles.

The RTL design description was done for all manufactured ASICs using the VHDL programming language followed by a functional verification in the Cadence nclaunch. The Synopsys Design Compiler (Version K-2015.06-SP2) was used to generate the gate-level netlist during the logic design step. Finally, the Cadence Encounter (for the ASIC2) and Innovus (for ASIC1 and ASIC3) were used for the physical layout during the place and route step. The layout was done for a maximum working frequency of 20 MHz (50ns clock cycle period). The usage of different software for the layout is due to the availability of the software. The first chip that was produced was ASIC2. Later, when we decided to manufacture the designs for ASIC1 and ASIC3 the Encounter software license was expired and replaced by Innovus in IHP.

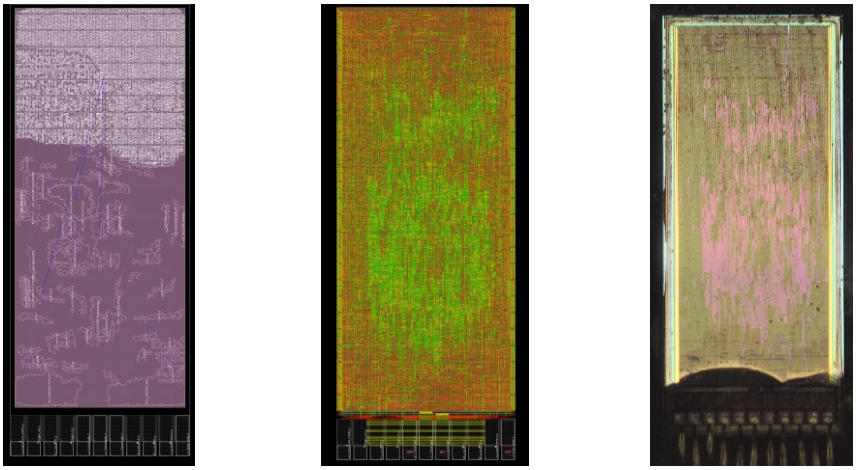
The chips were produced in the IHP 250 nm technology and bonded to printed circuit boards (PCB) without any packaging. The details about these ASICs as well as evaluation of their resistance against horizontal SCA attacks are provided in the next subsections.

7.1.1 Overview of manufactured ASICs

This subsection provides a short overview on the ASCIs investigated.

ASIC1. Design with a classical partial multiplier

This chip, i.e. ASIC1 is shown in Fig. 55. The chip is based on the IHP basic ***kP*** design and contains the partial multiplier for 59-bit long operands implemented according to the classical multiplication formula as described in section 6.1.1.1. The die has a size of 2758.56 µm x 1086.12 µm. The core size is 2437.16 µm x 1029.0 µm.



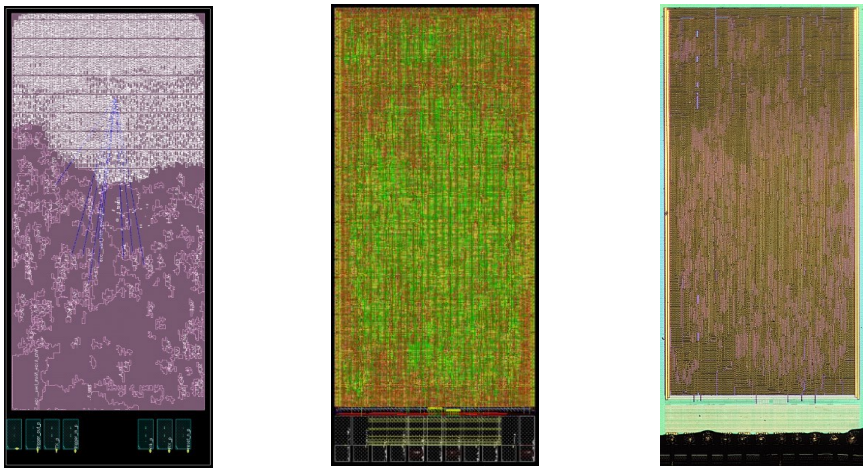
a) *b)* *c)*

Fig. 55. ASIC1: The placement of design blocks with a highlighted multiplier (*a*), the layout of the chip (*b*), a photo of the chip surface (c).

Please note, that the photos of the produced ASICs shown in Fig. 55-*c*, Fig. 56-*c* and Fig. 57-*c* are not in the same scale as the layout screenshots. The torn bottom edge on each of the photos is caused by a transparent epoxy resin molding compound used for the encapsulation of bonding wires to prevent their physical damage.

ASIC2. Design with a classical partial multiplier synthesized using the compile_ultra option

The second manufactured chip implements the same design as in ASIC1, i.e. it is the basic ***kP*** design with the classical partial multiplier for 59-bit long operands. The only difference to ASIC1 is the application of the *compile_ultra* option during the logic design step. As a result, the chip has a reduced design area and power consumption in comparison to ASIC1. The die has dimensions of 2458.36µm x 1086.12µm, the core of the chip has a size of 2136.96 µm x 1029.0 µm. Due to the same pads size in both ASICs the die size is almost the same, while the application of the *compile_ultra* option reduced the core area by about 12%. The layout screenshots of ASIC2 are shown in Fig. 56-*a* and Fig. 56-*b*. The highlighted white coloured area (Fig. 56-*a*) of the layout shows the shape and placement of the multiplier. Fig. 56-*c* shows a photo of the ASIC2's chip surface.



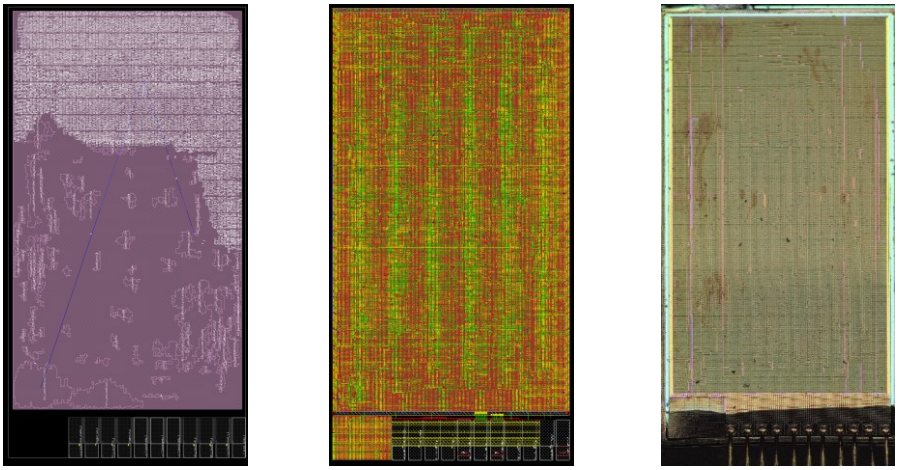
a) *b)* *c)*

Fig. 56. ASIC2: The placement of design blocks with a highlighted multiplier (*a*), the layout of the chip (*b*), a photo of the chip surface (c).

ASIC3. Design with a classical partial multiplier and a regular registers' addressing schedule

The third manufactured ASIC corresponds to the design described in section 6.3, i.e. the design with a classical partial multiplier (see section 6.1.1.1) and the regular scheduling of the block addressing (see section 6.1.3), synthesized using the standard *compile* option.

Due to the usage of additional logic cells required for the implementation of the regular scheduling this chip has the biggest area among all produced ones. The total die dimensions are 2758.24 µm x 1450.68 µm, whereas the core dimensions are 2436.84 µm x 1400.28 µm. The placement of design blocks with a highlighted multiplier is shown in Fig. 57-*a*. The layout of ASIC3 is shown in Fig. 57-*b*. Fig. 57-*c* shows a photo of ASIC3's chip surface. TABLE XIII. shows the main geometrical parameters of the produced ASICs.



a) *b)* *c)*

Fig. 57. ASIC3: The placement of design blocks with a highlighted multiplier (*a*), the layout of the chip (*b*), a photo of the chip surface (c).

TABLE XIII. PARAMETERS OF THE PRODUCED ASICS

Parameter:	**ASIC1**	**ASIC2**	**ASIC3**
Total area of Chip, µm^2	2996127.2	2670074.0	4001323.6
Chip Density (Total), %	91.661	91.345	91.143
Chip Density, % (Subtracting Physical Cells)	67.650	73.145	45.313
Total area of Core, µm^2	2507837.6	2198931.8	3412258.3
Core Density (Total), %	99.579	99.592	99.580
Total area of Standard cells, µm^2	2497273.6	2189956.6	3397929.7
Core Density, % (Subtracting Physical Cells)	70.892	77.493	45.838
Total area of Standard cells, µm^2 (Subtracting Physical Cells)	1777865.0	1704016.9	1564117.6
Total area of Pad cells, µm^2	249011.1	249011.1	249011.1

ASIC3 has the largest area of the chip/core, as can be seen in TABLE XIII. , while the area of the standard cells required to achieve the desired functionality is the smallest. This can be explained by the increased usage of physical cells that don't have any logical functionality and are required for the fulfilling of design rules during the place and route step as well as when finishing the chip.

7.1.2 **Analysis of the traces**

The analyzed traces represent the activity of the design during the execution of a single ***kP*** operation, which takes about 13000 clock cycles in our implementation.

The power traces after the synthesis and layout were obtained using the Synopsys PrimePower Version Q-2019.12-SP1 for an operating frequency of 4 MHz and a simulation step of 0.1ns, which results in 2500 simulated values per clock cycle.
For the measurements, i.e. collection of power and electromagnetic traces, the following equipment was used:

- SDG6032X 350MHz dual-channel Pulse/Arbitrary Waveform Generator
- R&S® HMP4040 high-performance power supply
- Langer FLS 106 IC Scanner 4-Axis Positioning System
- Riscure current probe
- Langer MFA-R 0.2-75 EM near-field probe
- LeCroy WavePro 254HD oscilloscope

The measurement setup is described detailed in section 3. Fig. 58 shows the placement of the Langer MFA-R 0.2-75 EM probe over the PCB board during the experiments.

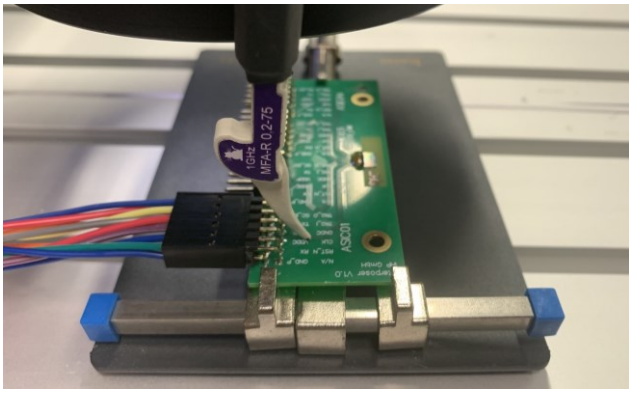


Fig. 58. The placement of the Langer MFA-R EM probe over the PCB board during the experiments.

The ASICs were connected to the 3.2V and 2.5V lines of the power supply to provide the voltage for the pads and core logic respectively. The current probe was connected to the core voltage line. The near-field probe was placed over the core voltage wire. Waveforms for the power and electromagnetic traces were captured with an oscilloscope at a sampling rate of 10 GS/s for ASIC1 and ASIC3 in order to obtain the same number of points per clock cycle as for the simulated traces. As for the traces for ASIC2, they were collected at a sampling rate of 2.5 GS/s[22].

During the simulation and measurements, the processed values (coordinates of the EC point $\boldsymbol{P}$ and the scalar k) were identical for all the investigated cases, the applied values are given in section 3.1.1. Therefore, the execution time for a single $k\boldsymbol{P}$ operation requires exactly the same number of clock cycles for all the designs investigated here.

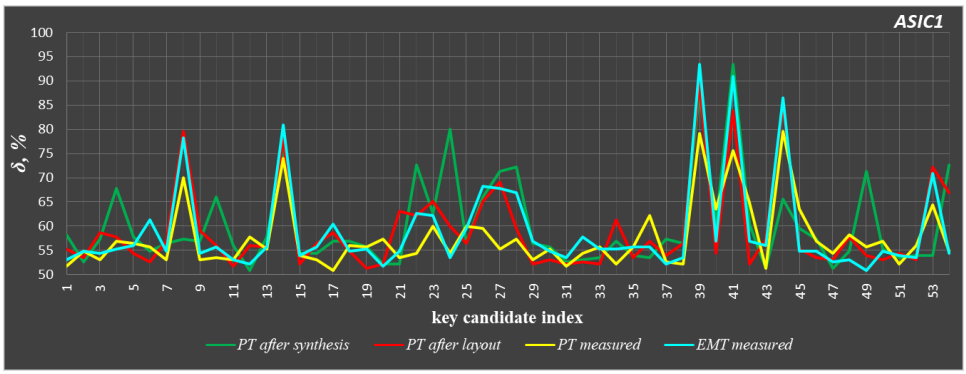
We applied several kinds of horizontal attacks either in the time and frequency domains, to investigate the SCA resistance of our designs. The results of the attacks are presented below.

7.1.2.1 Comparison to the mean attack

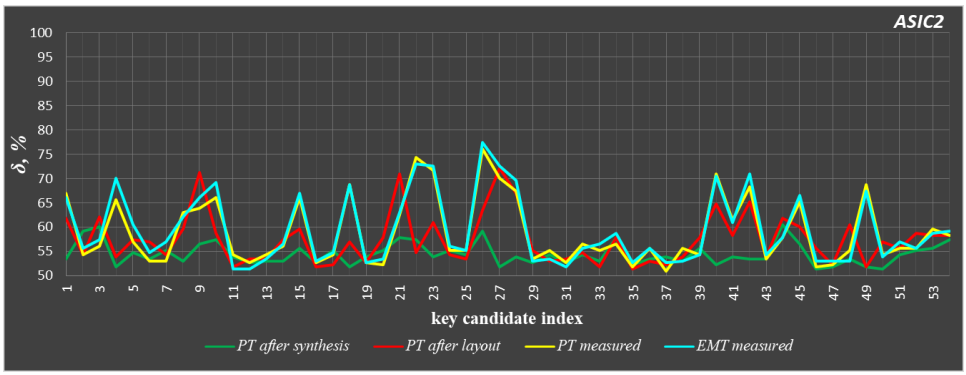
The comparison *to the mean* analysis was applied in total to 12 $k\boldsymbol{P}$ execution traces: – two simulated power traces (after the synthesis and after the layout, respectively), one measured

[22] The difference in the sampling rate applied for the traces measurements was caused due the long time distance between the experiments. However, this difference does not affect the fair chips comparison in terms of resistance evaluation. We evaluated this experimentally. We reduced the number of samples in a trace measured with the sampling rate of the 10 GS/s selecting only each 4th sample and analysed this "reduced" trace. We obtained the quite identical attack results.

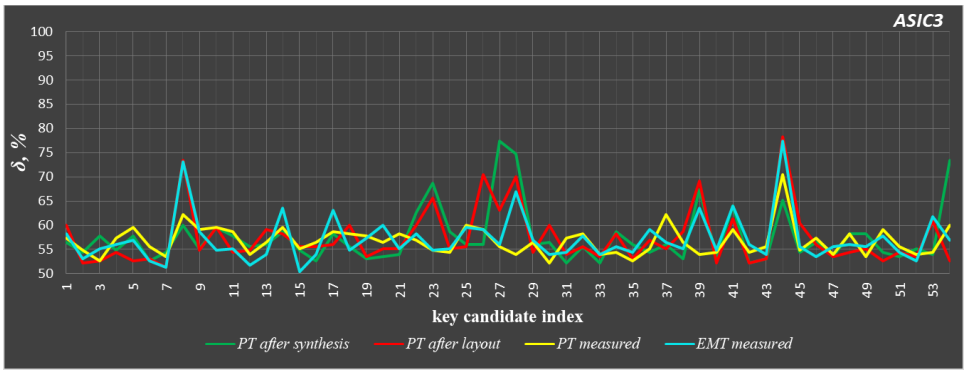
power trace and one measured electromagnetic trace for each of the three investigated ASICs. The attack was performed twice, i.e. for the compressed and raw traces, according to the description given in section 3.2.3. Primarily, the *comparison to the mean* attack using a compressed trace allows finding foremost strong leakage sources. On the other hand, applied to the raw traces it allows to find even a tiny leakage. The compression of the simulated traces was performed by representing each clock cycle with its average value (for more details see section 3.2.1). For the measured traces each clock cycle was represented by a sum of squared values according to section 3.2.1. The results of the comparison to the mean attack applied to the compressed traces are presented graphically in Fig. 59 and summarized in TABLE XIV.



a)



b)



c)

Fig. 59. Results of the comparison to the mean attack applied to the simulated compressed power traces after the synthesis (green line), after layout (red line), as well as to the compressed measured traces: the power trace (yellow line) and the electromagnetic trace (cyan line) for ASIC1 (a), ASIC2 (b) and ASIC3 (c).

As it can be seen in Fig. 59, there is a quite good correlation between expected results, i.e. those that were obtained by attacking power traces simulated after the layout (red lines), and the result for the measured power traces (yellow lines) of all the produced chips. Additionally, the attack results against each ASIC analysing its power and electromagnetic traces are similar, especially for ASIC2 (compare yellow and cyan lines in Fig. 59-*b*). For ASIC1 and ASIC3 the EMA attack is more successful compared to the power analysis attack (the peaks of the cyan lines in Fig. 59-*a* and Fig. 59-*c* are higher than the peaks of the yellow lines). TABLE XIV. summarizes the attack results for all 3 manufactured ASICs.

TABLE XIV. ATTACK RESULTS AGAINST MANUFACTURED ASICS

Attacked design	Analysed trace	Number of key candidates revealed with a correctness δ			δ_{max} ,%
		$70\% \le \delta < 80\%$	$80\% \le \delta < 90\%$	$\delta \ge 90\%$	
ASIC1	Synthesis	7	1	1	93.48
	Layout	3	3	0	89.56
	PT	4	0	0	79.56
	EMT	2	2	2	93.48
ASIC2	Synthesis	0	0	0	60.43
	Layout	3	0	0	72.17
	PT	4	0	0	76.09
	EMT	6	0	0	77.39
ASIC3	Synthesis	2	0	0	74.34
	Layout	2	0	0	78.69
	PT	1	0	0	70.43
	EMT	2	0	0	77.39

As follows from TABLE XIV. and Fig. 59, only the ASIC1 shows significant leakage that can be used in this kind of attack for successful revealing scalar *k*. Using data from the electromagnetic trace it was possible to obtain 4 key candidates with correctness higher than 80% (see the cyan line values for clock cycles 14, 39, 41, and 44 in Fig. 59-*a*). The maximum correctness is as high as 93.48% for the 39th key candidate. Attacking the power trace a leakage was noticed in the same four clock cycles as for the electromagnetic trace, however, the correctness of the key candidates was significantly lower. The maximum correctness in that case was 79.56% for the 44th key candidate. Thus, when applying the *comparison to the mean* attack to the compressed traces of the ASIC2 and ASIC3, either simulated or measured, all the obtained key candidates had a correctness of less than 80%. This means that the level of side-channel leakage is significantly lower than for ASIC1.

We also performed the comparison to the mean attack against measured uncompressed – i.e., raw – traces. The results are presented in TABLE XV.

TABLE XV. RESULTS OF THE COMPARISON TO THE MEAN ATTACK ANALYSING RAW (UNCOMPRESSED) TRACES

Attacked design	Analysed trace	Number of key candidates revealed with a correctness δ			δ_{max} ,%
		$70\% \le \delta < 80\%$	$80\% \le \delta < 90\%$	$\delta \ge 90\%$	
ASIC1	Synthesis	5000	1250	625	92.17
	Layout	3125	1875	0	88.26
	PT	1844	468	13	92.17
	EMT	1443	177	4	91.74
ASIC2	Synthesis	0	0	0	60.87
	Layout	1875	0	0	72.17
	PT	37	0	0	76.96
	EMT	108	1	0	80.87
ASIC3	Synthesis	130	120	183	100*
	Layout	410	250	231	100**
	PT	344	0	0	76.09
	EMT	115	0	0	78.26

*23 times
**22 times

For the ASIC1 it was possible to find key candidates with a correctness of up to 92.17% analysing the raw power trace, whereas the analysis of the same compressed trace resulted in a maximum correctness of 79.56%. For the simulated traces, as well as for the measured electromagnetic trace, the results for analysing compressed and raw traces are comparable in terms of the maximum correctness δ_{max}. Likewise, for ASIC2 there is no significant difference between the compressed and raw traces analysis results.

As for the ASIC3, it was possible to find 23 and 22 key candidates with a correctness of 100% out of 135000 key candidates by analysing uncompressed simulated power traces after synthesis and layout respectively. Please note, that the maximum correctness obtained analysing the compressed version of the traces was only 74.34% for the power trace after synthesis and 78.69% for the power trace after layout (see TABLE XIV.). It might seem that such results put an end to this design. However, it is always necessary to remember that the compression of the traces can influence significantly the attack results, i.e. it can make the "distinguishers" between '0' and '1' shapes more visible or – opposite – it can hide them. Due to this knowledge, designers have to analyse the compressed as well as the row traces with the goal to select the appropriate compression. An additional important fact is that the measured traces are not noise-free, opposite to the simulated traces. The noise can hide some SCA leakage sources. This can explain, why the results of the compressed, as well as the uncompressed measured traces, are similar. The attack success analysing the measured uncompressed traces is slightly higher than for the compressed ones, in our experiments.

Attack results analysing compressed as well as uncompressed measured traces of the ASIC3 show that the applying of the regular block addressing increases the design resistance against the horizontal *comparison to the mean* attacks. Out of 135000 key candidates, there was no one revealed with the correctness higher than 76.09% analysing the measured power trace and 78.26% analysing the measured electromagnetic trace.

7.1.2.2 FFT-based attack against power and electromagnetic traces measured on ASICs

As a next step, we applied an attack conducted in the frequency domain, according to the description given in section 3.2.5.2. We attacked 6 traces of the $k\boldsymbol{P}$ execution – one measured power trace and one measured electromagnetic trace for each of the three investigated ASICs.

Fig. 60 shows the attack results for the PT and EMT traces measured during $k\boldsymbol{P}$ execution on ASIC1.

The best key candidate, obtained by attacking the power trace of ASIC1, has a correctness of 86,96%. The 16 key candidates revealed with a correctness of more than 80% were obtained analysing harmonics 3, 4, 7, 9, 12, 13, 22, 24, 26 and 27.

In the case of the attack against the EM trace of the ASIC1, two key candidates were revealed with a correctness of more than 90% and 8 more key candidates with a correctness in the range between 80% and 90 %. The best key candidate has a correctness of 93.04%. These 10 key candidates were obtained by the analysis of harmonics 2, 3 and 4 (see Fig. 60-*c* and Fig. 60-*d*).

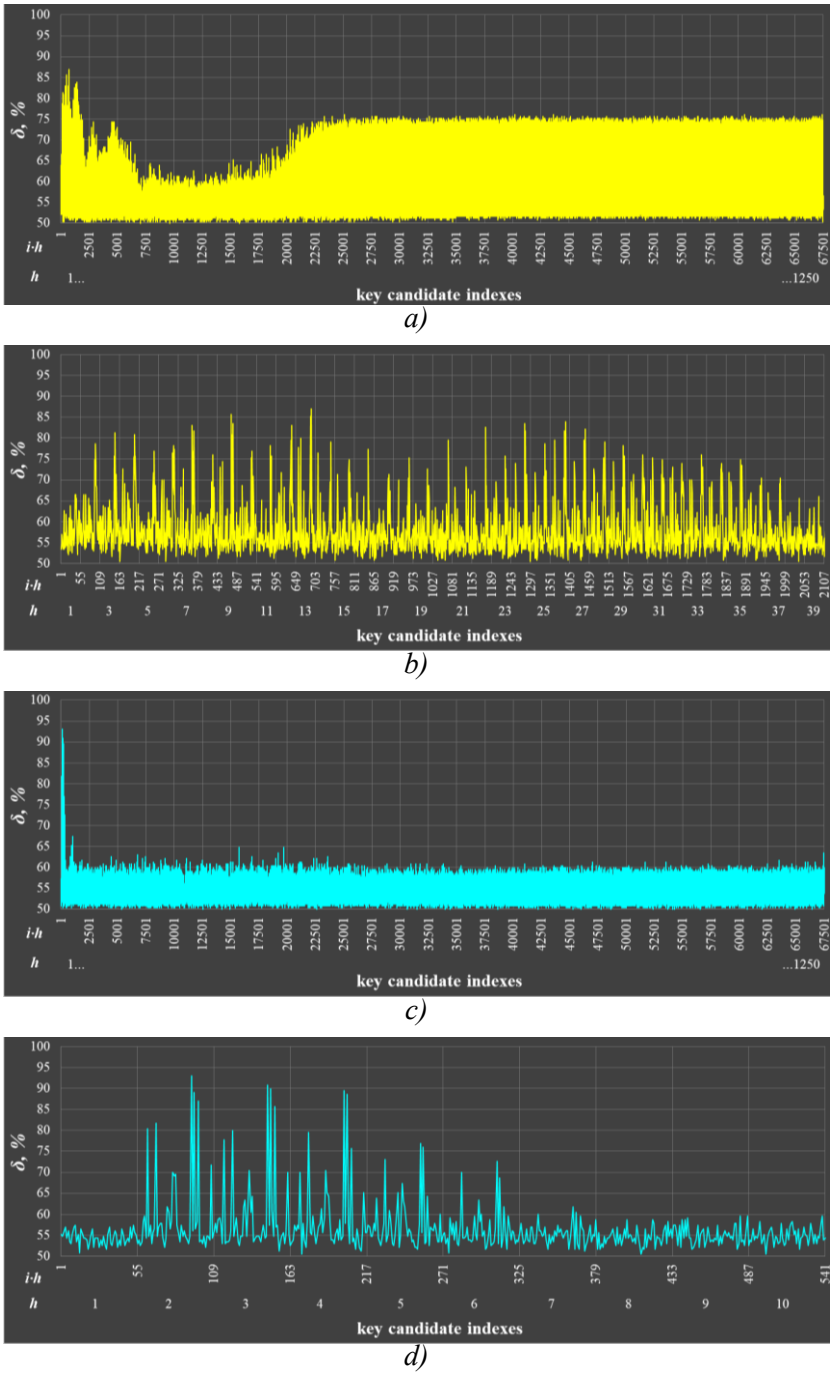
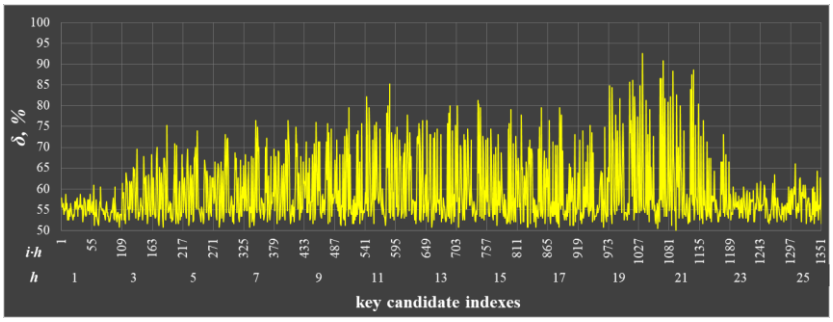
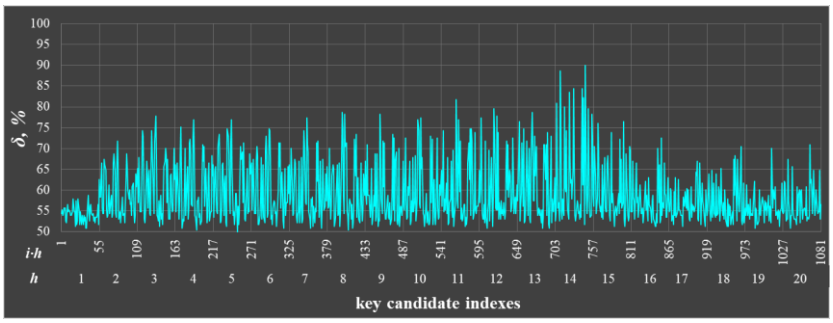


Fig. 60. Graphical representation of the results of the FFT-based horizontal attack against ASIC1. Figures (a) and (c) show the correctness for all 67500 key candidates obtained by attacking power and electromagnetic traces, respectively. Figures (b) and (d) show a part of the results, obtained by attacking power and electromagnetic traces respectively, for the harmonics containing key candidates revealed with the highest correctness.

Fig. 61 shows the attack results for the PT and EMT traces measured during $k\boldsymbol{P}$ execution on ASIC2. Key candidates with a correctness of more than 80% are obtained predominantly by the analysis of harmonics 11, 14, 19 to 21 for the power trace. For the electromagnetic trace the same level of correctness can be achieved using the analysis of harmonics 11 and 14. The best key candidates attacking the power and the electromagnetic trace have a correctness of 92.61% and 90.00% respectively. A more detailed investigation for ASIC2 as well as an approach allowing to increase the success rate of the conducted attack are described in [29].



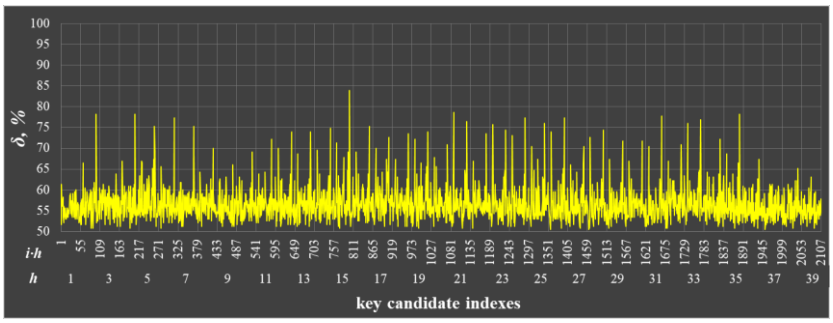
a)



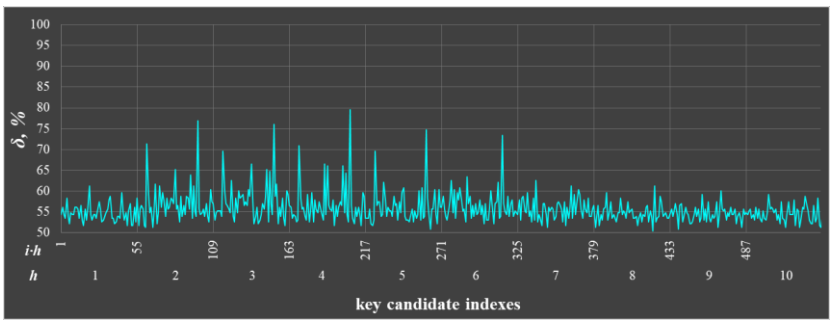
b)

Fig. 61. Graphical representation of the results of the FFT-based horizontal attack against ASIC2. Figures (a) and (b) show a part of the results, obtained by attacking power and electromagnetic traces respectively, for the harmonics containing the key candidates revealed with the highest correctness.

Finally, Fig. 62 shows the attack results for the traces measured on ASIC3.



a)



b)

Fig. 62. Graphical representation of the results of the FFT-based horizontal attack against ASIC3. Figures (a) and (b) show a part of the results, obtained by attacking power and electromagnetic traces respectively, for the harmonics containing the key candidates revealed with the highest correctness.

Attacking the traces, only one key candidate was revealed with a correctness higher than 80%. The best key candidates with a correctness of 83,91% and 79,56% were obtained analysing power trace and electromagnetic trace respectively.

TABLE XVI. summarize the results of the attack performed against three ASICs. The detailed location of the key candidates revealed with a certain level of correctness within the slot is given in TABLE XVII.

TABLE XVI. RESULTS OF THE FFT-BASED ATTACK AGAINST THE THREE ASICS

Attacked design	Analysed trace	Number of key candidates revealed with a correctness δ			δ_{max} ,%
		$70\% \le \delta < 80\%$	$80\% \le \delta < 90\%$	$\delta \ge 90\%$	
ASIC1	PT	1803	16	0	86.96
	EMT	11	8	2	93.04
ASIC2	PT	128	23	2	92.61
	EMT	99	7	1	90.00
ASIC3	PT	45	1	0	83.91
	EMT	7	0	0	79.56

TABLE XVII. DETAILS FOR THE FFT-BASED ATTACK AGAINST THE THREE ASICS

Clock cycle index within the slot	ASIC1					ASIC2						ASIC3		
	PT		EMT			PT			EMT			PT		EMT
	Number of key candidates revealed with a correctness δ													
	$70\% \le \delta < 80\%$	$80\% \le \delta < 90\%$	$70\% \le \delta < 80\%$	$80\% \le \delta < 90\%$	$90\% \le \delta < 100\%$	$70\% \le \delta < 80\%$	$80\% \le \delta < 90\%$	$90\% \le \delta < 100\%$	$70\% \le \delta < 80\%$	$80\% \le \delta < 90\%$	$90\% \le \delta < 100\%$	$70\% \le \delta < 80\%$	$80\% \le \delta < 90\%$	$70\% \le \delta < 80\%$
3						5	4		1	1				
4						7			5					
8	**4**		**1**	**1**		**7**	**2**	**1**	**4**	**1**		**1**		**2**
9	**3**					**4**			**5**			**11**		
10						**3**			**2**					
14	11		3	1		5	2		1					
15	5					1			1					
17						**5**			**1**					
18						**5**								
21						6	1		3	1				
22						9			9	1				
23						6			6					
26			**2**			**8**			**11**					
27						**10**			**8**	**1**				
28						**1**			**5**					
39	848	4	2	1	2	4	4		3	1				
40	15					5			10					
41	23	4	1	3		7	3			1				
42	10	2				5			9					
43	**1**													
44	**881**	**6**	**1**	**2**		**8**	**3**	**1**	**5**		**1**	**7**	**1**	**5**
45	**2**								**1**			**26**		
48						5	2		1					
49						6			5					
53			**1**			**5**	**2**		**2**					
54						**1**			**1**					

Please note that the resistance of ASIC2 attacked using the comparison to the mean method seems to be high, i.e. only one key candidate is extracted with a correctness of about 81% analysing the uncompressed electromagnetic trace (see TABLE XV.). But using FFT for the analysis of the same uncompressed traces we extracted one key candidate with a correctness of 90% attacking when attacking the EMT, two key candidates with the correctness higher than 90% attacking the PT, and much more key candidates with a correctness between 70% and 80%, i.e. the success of the FFT-attacks is a higher than the one of the comparison to the mean.

The resistance of the ASIC3 seems to be high for both kinds of analysis, i.e. when using the comparison to the mean as well as when using FFT. Please note that in clock cycles 7-10, 16-19, 26-28, 35-37, 43-46 and 53-54-1 the regular addressing schedule is not applicable (see section 6.1.3), i.e. in these clock cycles an increase in the resistance was not expected. These clock cycles are marked in **bold** in TABLE XVII. . Thus, the high attack success in the clock cycles 8 and 44 for the ASIC 3 was expected.

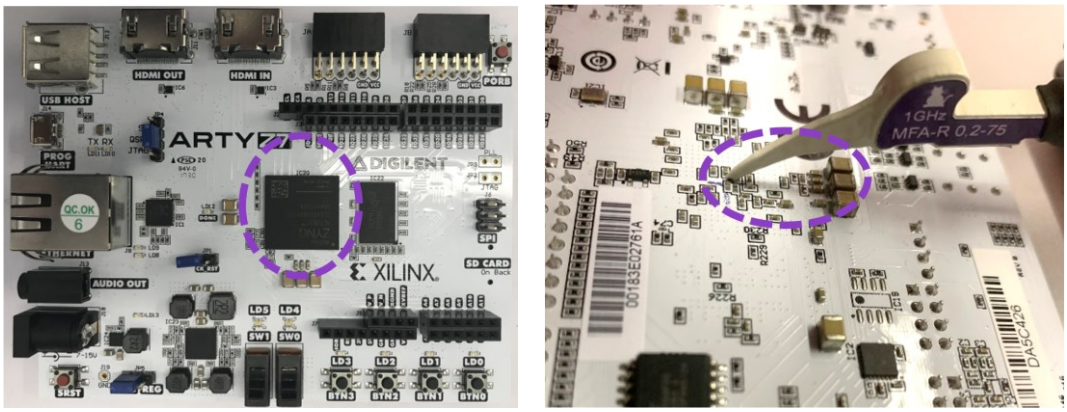
7.2 FPGA implementation

FPGAs are an interesting alternative to ASICs at least for niche markets. The application of FPGAs also can help designers to quickly evaluate their ideas in prototyping as well as validate a design or concept on a real device. Therefore, it is of some importance to know how the target platform influences the vulnerability of an implementation.

We ported all 3 designs selected for manufacturing as ASICs (see section 7.1.1) to an FPGA, i.e. we used the same VHDL code for implementation in ASICs as well as in FPGA. For the FPGA implementation of the ASIC1 and ASIC 3 designs the default synthesis and implementation strategies were applied in the Vivado Design Suite software. These implementations are further denoted as FPGA1 and FPGA3 designs respectively. As for the FPGA implementation of the ASIC2 design, further denoted as FPGA2, the "Flow_PerfOptimized_high" synthesis strategy and "Performance ExplorePostRoutePhysOpt" implementation strategy were applied with the goal to increase the design's performance. Please note, that the applied strategies for the FPGA2 design are not equal to the optimization option ("*compile_ultra*") applied for the ASIC2 design. However, we tried to achieve a similar effect, i.e. increase the design performance.

As target platform we took an Arty Z7-20 development board equipped with a Xilinx 7-series FPGA produced in a 28 nm technology. The attacked FPGA is marked with a circle in Fig. 63-*a*. The board used is not well suitable for the connection of a current probe. Therefore, instead of measuring power traces that require a board modification, we applied a near-field magnetic probe to capture electromagnetic traces. As clock source we decided to use an external signal generator. This allowed us to apply a clock frequency of 4 MHz with the goal to capture the traces under the same operating conditions as in the experiments with the ASICs. The minimum possible operating frequency for the internal clock generator supported by the board is 4.768 MHz.

The electromagnetic traces were captured during the execution of $k\boldsymbol{P}$ operations at a 10 GS/s sampling rate by a LeCroy WavePro 254HD oscilloscope and a Langer MFA-R 0.2-75 near-field probe connected to it. We placed the MFA-R probe close to one of the power decoupling capacitors. The exact location of the probe is given in Fig. 63-*b*.



a) *b)*

Fig. 63. Arty Z7-20 Development board (a) and the measurement place on its backside (b).

7.2.1 **Evaluation of the proposed design improvements**

In this section we evaluate the side-channel resistance of our designs against horizontal SCA attacks using the same approaches as we applied for the manufactured ASICs.

7.2.1.1 Comparison to the mean attack against electromagnetic traces measured on FPGA

We applied the *comparison to the mean* attack to the electromagnetic trace of the ***kP*** execution measured for each of the three designs ported to the FPGA. As in the cases of the ASICs, the attack was performed for the compressed as well as uncompressed traces, according to the description given in section 3.2.3. We applied the same compression method as for the ASIC traces. The results of the *comparison to the mean* attack applied to the compressed traces are presented graphically in Fig. 64.

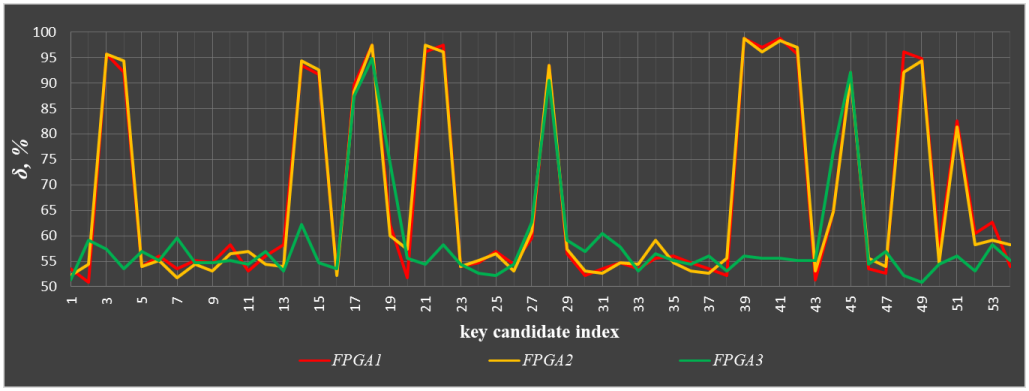


Fig. 64. Results of the comparison to the mean attack applied to the measured electromagnetic trace of FPGA1 design (red line), FPGA2 design (yellow line), as well FPGA3 design (green line).

TABLE XVIII. summarizes the attack results for the three investigated FPGA designs.

TABLE XVIII. RESULTS OF THE COMPARISON TO THE MEAN ATTACK AGAINST COMPRESSED TRACES

Attacked Design	Number of key candidates revealed with a correctness δ			δ_{max} ,%
	$70\% \le \delta < 80\%$	$80\% \le \delta < 90\%$	$\delta \ge 90\%$	
FPGA1	0	2	15	98,70
FPGA2	0	2	15	98,70
FPGA3	2	1	3	94,78

We also performed the *comparison to the mean* attack against uncompressed FPGA traces, see results in TABLE XV.

TABLE XIX. RESULTS OF THE COMPARISON TO THE MEAN ATTACK AGAINST UNCOMPRESSED TRACES

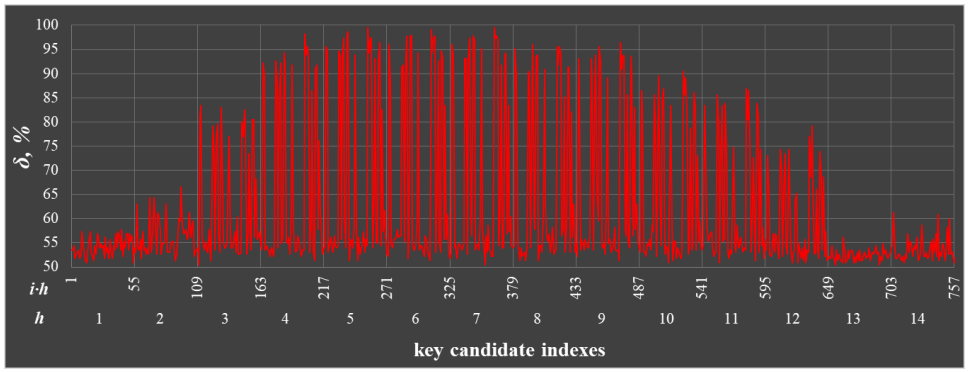
Attacked Design	Number of key candidates revealed with a correctness δ			δ_{max} ,%
	$70\% \le \delta < 80\%$	$80\% \le \delta < 90\%$	$\delta \ge 90\%$	
FPGA1	2980	2543	1276	99,13
FPGA2	2900	2461	1306	100
FPGA3	764	480	126	93,91

As it can be seen from TABLE XVIII. and TABLE XIX. , results for compressed and uncompressed trace analysis are quite similar in terms of the maximum correctness δ_{max} for all three investigated FPGA designs. In comparison to the results obtained by attacking electromagnetic traces of the ASICs (see section 7.1.2.1), the number of key candidates revealed with extremely high correctness increased significantly. This holds true for the attack carried out against compressed as well as uncompressed traces. Additionally, our evaluation revealed that the impact of the compiler options for FPGAs is by far smaller than in the case of ASICs.

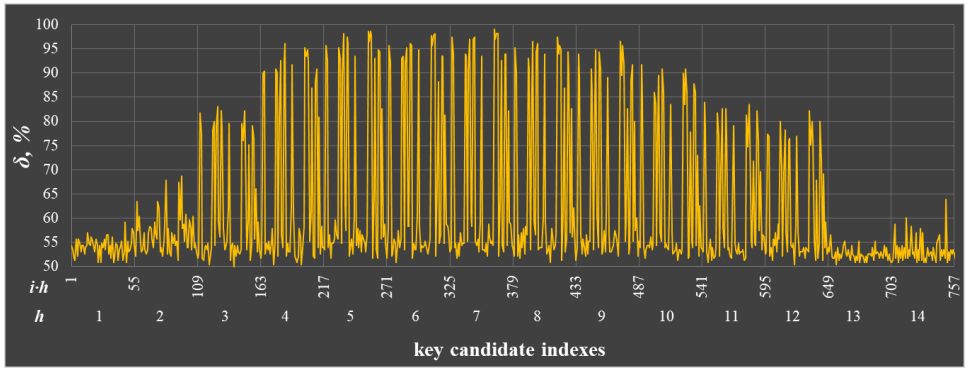
Attack results analysing the measured trace of the FPGA3 show that the applying the regular blocks' addressing increases the design resistance against the horizontal *comparison to the mean* attacks. In the case of the FPGA3, it can be observed as a reduced number of key candidates revealed with a high level of correctness in comparison to the other two FPGA designs. However, the attack success rate is by far higher than for the ASIC3. Please note that in our designs the addressing of blocks cannot be changed in all clock cycles due to the implemented logic. This holds true for the following sequences of clock cycles: 7-10, 16-19, 26-28, 35-37, 43-46, 53-54-1. For the FPGA3 design, the five key candidates with the highest correctness are 17, 18, 28, 44 and 45, i.e. they are part of the clock cycle sequences that cannot be modified. Due to this fact a significant reduction of the success rate of the attacks in these clock cycles was not expected. But the effectiveness of the regular blocks'/registers' addressing schedule for an FPGA implementation can be deduced by examination of the following clock cycles: 3-4, 14-15, 39-42, 48-49 and 51: here are in contrast to the other FPGA designs no key candidates with a correctness of more than 60% could be revealed.

7.2.1.2 FFT-based attack against electromagnetic traces measured on FPGA

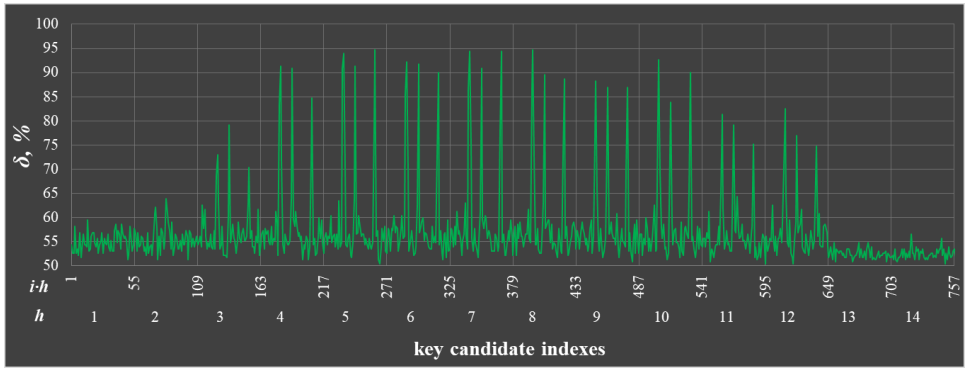
We performed our attack in the frequency domain against three electromagnetic traces of the *investigated k**P*** designs measured on the FPGA, according to the description given in section 3.2.5.2 (i.e. in exactly the same way as for the ASICs). Results of the attacks for all 3 designs are partially represented in Fig. 65, i.e. for the key candidates $k^{candidate\ h,i}$ with the highest correctness.



a)



b)



c)

Fig. 65. Results of the FFT-based attack against electromagnetic traces for FPGA1 (a), FPGA2 (b) and FPGA3 (c).

Fig. 65-*a*, -*b* and -*c* show results obtained attacking the first 13 harmonics of the measured traces for the FPGA1, FPGA2 and FPGA3 designs, respectively. All the key candidates revealed with a correctness higher than 75% were extracted using the information from harmonics 3 to 12. The correctness of the key candidates obtained from the rest of the harmonics is not high enough and therefore may not be taken into account as a source of significant leakage.

Applying FFT-based horizontal attack to electromagnetic traces of the $k\boldsymbol{P}$ execution for FPGA1, FPGA2 and FPGA3 designs the maximum achieved correctness was as high as 99.57%, 99.13% and 94.78% respectively.

An attack results summary is given in TABLE XX.

TABLE XX. RESULTS OF THE FFT-BASED ATTACK AGAINST FPGA DESIGNS

Attacked Design	Number of key candidates revealed with a correctness δ			δ_{max} ,%
	70% ≤ δ < 80%	80% ≤ δ < 90%	δ ≥ 90%	
FPGA1	30	49	81	99,57
FPGA2	38	47	81	99,13
FPGA3	18	16	13	94,78

TABLE XXI. provides information regarding the location within the slot for the key candidates from TABLE XX.

TABLE XXI. DETAILS FOR THE FFT-BASED ATTACK AGAINST FPGA DESIGNS: LOCATION OF THE KEY CANDIDATES IN THE SLOT

Clock cycle index in the slot	**FPGA1**			**FPGA2**			**FPGA3**		
	Number of key candidates revealed with a correctness δ								
	70% ≤ δ <80%	*80%≤ δ <90%*	*90%≤ δ <100%*	*70%≤ δ <80%*	*80%≤ δ <90%*	*90%≤ δ <100%*	*70%≤ δ <80%*	*80%≤ δ <90%*	*90%≤ δ <100%*
3	2	3	6	1	3	6	0	0	0
4	1	6	2	3	3	4	0	0	0
14	2	2	6	2	2	6	0	0	0
15	2	4	3	2	3	4	0	0	0
17	3	4	2	2	5	2	2	5	1
18	2	2	6	1	3	6	1	3	6
19	0	0	0	0	0	0	4	0	0
21	1	3	6	1	2	7	0	0	0
22	2	1	6	3	1	6	0	0	0
28	2	2	5	3	2	5	3	3	4
39	2	1	7	1	3	6	0	0	0
40	1	2	6	3	2	5	0	0	0
41	1	3	6	2	2	7	0	0	0
42	2	1	6	3	1	6	0	0	0
44	0	0	0	0	0	0	5	0	0
45	3	3	3	3	4	2	3	5	2
48	1	3	6	3	4	4	0	0	0
49	0	4	5	3	2	5	0	0	0
51	3	5	0	2	5	0	0	0	0

As it can be seen from TABLE XXI. and Fig. 64, the key candidates revealed with a correctness higher than 90% have the same indexes within the slot as in the case of the comparison to the mean attack. Thus, different analysis methods applied in attacks point to the same leakage sources.

7.3 Dependence of the FPGAs and ASICs resistance on the frequency

In additional investigations, we experimented with the designs denoted as FPGA1 and FPGA2 in previous sections attempting to identify the maximum of its operating frequency with the goal to estimate their performance and resistance.

For the Arty Z7-20 board, ***kP*** results for the FPGA1 design were still correctly calculated for the frequencies slightly above 200 MHz, i.e. the default synthesis/implementation strategies allow to get the design with a maximum operating frequency of 200 MHz.

The application of the "Flow_PerfOptimized_high" synthesis strategy and "Performance_ExplorePostRoutePhysOpt" implementation strategy together are performance-optimizing compile options. We applied them for the synthesis of the FPGA2 design. It allowed us to increase the operating frequency to 240 MHz. The design operating at this frequency is capable to perform about 18500 scalar multiplications per second for the elliptic curve *B-233* (without the input/output data transmission). However, after capturing the corresponding electromagnetic trace we noticed, that the secret scalar k can be extracted using only visual inspection of the trace, i.e. with pure eyes (see Fig. 17 in section 3.2.5.1).

Due to the fact that our design – i.e., the VHDL code implementing the Montgomery ladder – is highly regular, the successful simple analysis attack was not expected and was highly surprising. After a successful SEMA attack, we decided to evaluate the resistance of the FPGA2 design, i.e. the same VHDL implementation but synthesized using the performance-optimization compile options for five additional frequencies – 10, 50, 100, 160 and 200 MHz. The choice of the frequencies was done with respect to the possibility of getting an integer number of captured samples per clock cycle period, as the oscilloscope has a fixed number of available settings for the sampling rate. This aspect has huge importance when conducting SCA attacks.

For each of the investigated frequencies, we synthesized the corresponding design. For each of the designs, we captured an electromagnetic trace during a $k\boldsymbol{P}$ execution (always with the same input data) using the near-field probe MFA-R 0.2-75 from Langer and a LeCroy HDO9404-MS oscilloscope with a maximum sampling rate of 40 GS/s. The measurement place for the board is shown in Fig. 63-*b*. When capturing the traces we used different sampling rates for different designs in order to assure a fair assessment of the attack results. This led to a similar amount of samples captured per clock cycle for all frequencies.

Approximate times of the $k\boldsymbol{P}$ executions and the parameters of the measurement setup are given in TABLE XXII.

TABLE XXII. MEASUREMENT PARAMETERS AND *kP* EXECUTION TIME

Design frequency, MHz	**Sampling rate, GS/s**	**Samples per clock cycle period**	**Approximate *kP* execution time, µs**	**Number of *kP* operations per second**
10	2.5	250	1300	769
50	10	200	260	3846
100	20	200	130	7692
160	40	250	81	12345
200	40	200	65	15384
240	40	~166.7	54	18518

The screenshots of the captured traces for the six investigated frequencies for the FPGA2 designs are shown in Fig. 66. The zoomed-in part on each oscillogram represents a fragment of a trace in which about 9 key bits are processed in the main loop (this part is slightly highlighted in the oscillogram with a full $k\boldsymbol{P}$ trace, at its beginning). As it can be seen in Fig. 66, the shape of the measured trace, as well as its amplitude, depend significantly on the operating frequency.

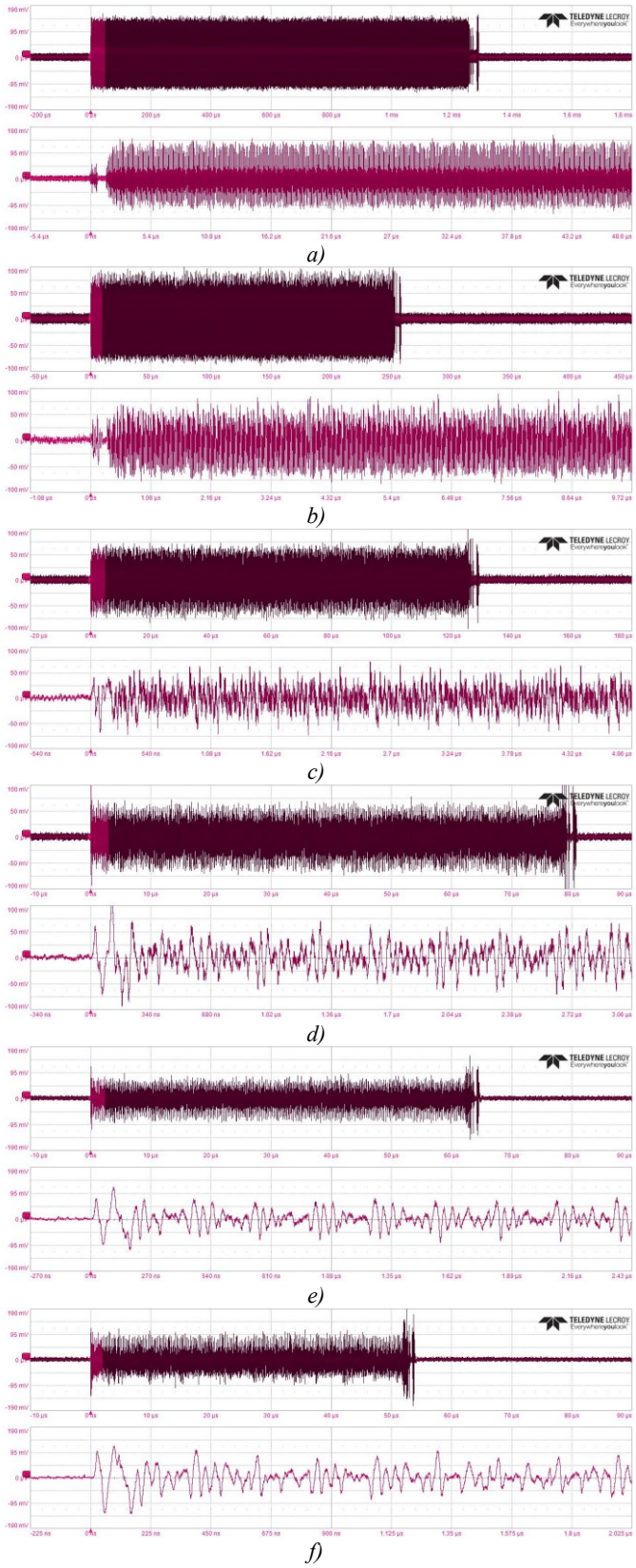


Fig. 66. Screenshots of the captured traces of the *k**P*** execution of the designs FPGA2 running at 10 MHz *(a)*, 50 MHz *(b)*, 100 MHz *(c)*, 160 MHz *(d)*, 200 MHz *(e)*, 240 MHz *(f)* clock frequencies.

TABLE XXIII. shows the results of our automated simple attack performed against each design synthesized applying performance-optimizing compile options, i.e. against FPGA2 designs, synthesised for different frequencies.

TABLE XXIII. RESULTS OF AUTOMATED SEMA ATTACK AGAINST FPGA2 DESIGNS (PERFORMANCE-OPTIMIZED)

Frequency, MHz	**samples per slot: *N***	**number of samples with „gaps“, i.e. number of key candidates with correctness δ=100%**		**max „gap“ distance, V**
		as a number ***n***	as a relative number ***n/N***, %	
10	13500	0	0 %	0
50	10800	40	0.37 %	0.00616
100	10800	126	1.17 %	0.00983
160	13500	296	2.19 %	0.00451
200	10800	192	1.78 %	0.00274
240	9000	304	3.38 %	0.01649

Starting from the 50 MHz clock cycle frequency we were able to reveal the used scalar k completely. The number of the key candidates with the correctness of 100% increases with the increasing frequency. For the frequency of 50 MHz, we extracted 40 key candidates that are identical to the key while for the frequency of 240 MHz the number of key candidates with such a correctness increased to 304. Please note that an attack is successful even if only 1 key candidate has a correctness of 100%.

We performed the attack described above also against traces measured for design FPGA1, i.e. the one with default synthesis/implementation strategies, for clock frequencies up to 100 MHz to confirm that the attack success was not caused by the applied compile strategies and does not significantly depend on them. TABLE XXIV. shows the results of our automated simple attack performed against each design synthesized with the default options.

TABLE XXIV. ATTACK RESULTS FOR FPGA1 DESIGNS (WITH DEFAULT COMPILE OPTIONS)

Frequency, MHz	**samples per slot: *N***	**number of samples with „gaps“, i.e. number of key candidates with correctness δ=100%**		**max „gap“ distance, V**
		as a number ***n***	as a relative number ***n/N***, %	
10	13500	0	0 %	0
20	13500	0	0 %	0
40	13500	7	0.05 %	0.00428
50	10800	18	0.17 %	0.00368
80	13500	78	0.58 %	0.01053
100	10800	107	0.99 %	0.00554

Fig. 67 represents graphically the data given in TABLE XXIII. and TABLE XXIV. , i.e. it shows the number of key candidates with a correctness δ=100% for all designs attacked.

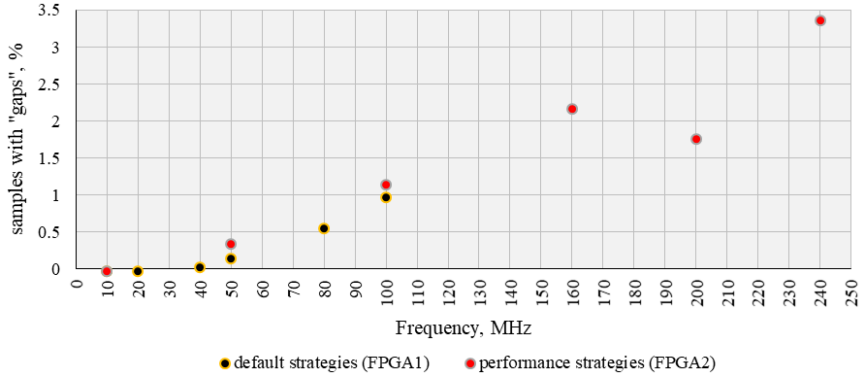


Fig. 67. Graphical representation of the attack results given in TABLE XXIII. and TABLE XXIV. (the “default strategies” refer to FPGA1 designs and the “performance strategies” refer to FPGA2 designs)

In order to investigate if the vulnerability of the design depending on the target frequency is observable also on other platforms than FPGAs, we synthesized the IHP basic ***kP*** design[23] for different frequencies using gate libraries for two different IHP CMOS technologies using the simple *compile* option that is comparable to the "default strategies" compiling options applied for FPGA1 designs . Applying the *compile* option the area of the ASIC designs was optimized. We synthesized our design for 16 different frequencies for the IHP 250 nm technology and for 20 frequencies for the IHP 130 nm technology and analysed the 16+20=36 simulated power traces. We decided to generate and analyse so many different versions to ensure a fair and detailed assessment of the frequency-dependent vulnerability. Fig. 68 shows the relative number of key candidates with a correctness δ=100% for all attacked ASIC designs. As the maximum relative number of samples with "gaps" for 130 nm technology is less than 1%, we used different scales of the axis for the 250 nm and the 130 nm technology, respectively.

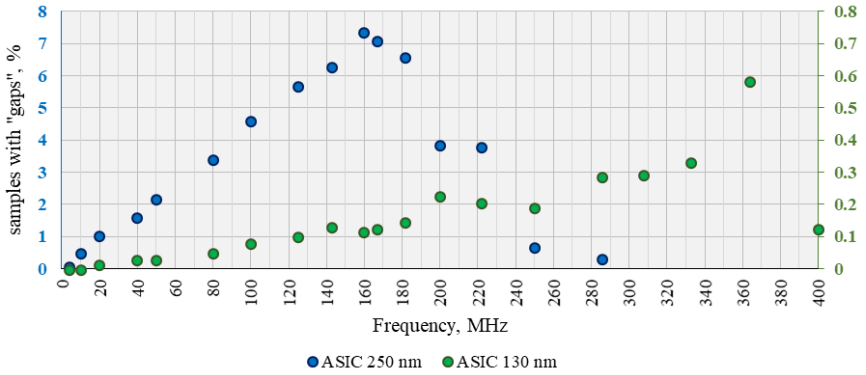


Fig. 68. Graphical representation of the attack results against ASIC designs

The interesting fact is that the correctness of keys revealed shows the same trend for the FPGA and ASIC designs in the 250 nm technology in the frequency range of up to 200 MHz[24]. The number of points in an analysed trace, which are strong leakage sources allowing to reveal the key, is increasing for the frequencies up to 160 MHz and starts to decrease in the frequencies interval from 160 to 200 MHz. For the 130 nm technology the maximum number of such points is achieved at 366MHz frequency and is decreasing significantly at 400 MHz.

The reduced number of points representing strong leakage for the 130 nm technology compared to the 250 nm technology may be explained not only by the use of different technologies but also by the fact, that 130 nm technology is significantly faster, i.e. the time in each clock cycle that corresponds to the active gate switching is by far shorter for the 130 nm technology than the one for the 250 nm. Therefore, the effective number of simulated values where the gates are switching is less for the 130 nm than for the 250 nm technology.

We observed a similar trend for ASICs as for the FPGA implementation (see Fig. 68 and Fig. 67): for very low frequencies only a few points in the analysed traces represent strong leakage sources allowing to reveal the key successfully. For the "middle" frequencies the number of points of the traces that allow to successfully reveal the key increases with increasing frequency. In contrast to FPGA, the attacks using ASICs simulations show that the vulnerability of designs synthesised for very high frequencies is not high.

Results shown in TABLE XXIII. , TABLE XXIV. as well as Fig. 67 and Fig. 68 should alarm all designers: the Montgomery ***kP*** algorithm that is due to its regularity in the literature declared as resistant against simple SCA attacks is not resistant against simple automated

[23] The design is denoted as *design1* in section and differs from the design 2 only in its partial multiplier

[24] We are aware that FPGA and ASIC designs are are not the same: their partial multipliers are different. The more interesting is the fact that the clock signal frequency affects the resistance of the designs similarly.

analysis for almost all investigated frequencies. This is true for different target platforms, i.e. for an FPGA and for ASICs synthesized for two different technologies. This is true even for our improved implementation of the Montgomery ladder, i.e. for the *design2*. In this design:

- the classical partial multiplier is the source of the increased noise, whereby the field multiplier is active in all clock cycles during main loop iterations;
- all operations including the addressing of blocks/registers are implemented in parallel to field multiplications.

Important is the fact that the resistance of the same implementation, i.e. the same VHDL code, depends significantly on the target frequency for each here investigated target platform. This fact has to be taken into account, i.e. designers have to evaluate the resistance of their implementations for each target frequency in order to know for sure how vulnerable their implementations are. One of the possible attack scenarios is when attackers can try to run the designs using an increased clock frequency. If it is possible, it can reduce / or negatively impact the design's resistance.

7.4 Summary

Here the main results of the experiments described in this section are summarized:

- The regular addressing schedule proposed in 6.2 is an effective means to reduce the success of horizontal address-bit SCA attacks if it may be applied. Due to the algorithm's data flow this may not be possible in all clock cycles).
- FFT-based horizontal attacks, as well as autoSPA attacks, are effective analysis methods that can be used by designers for the evaluation of the resistance of cryptographic accelerators.
- The resistance of cryptographic designs depends significantly on the processing frequency. This means that designers have to evaluate the resistance of their implementations for the target frequency and implement mechanisms that prevent attackers from running the designs at any other frequency.
- The SCA attack results obtained using the simulated PTs (after layouts, see Fig. 59) are similar to the results obtained analysing the measured PT of the manufactured ASICs, i.e. the simulated PT can be used for the evaluation of the design resistance against SCA attacks in a relatively early design phase.

We currently have no explanation for the fact that the vulnerability of the Montgomery ladder increases with the execution frequency. But our analysis of measured and simulated traces clearly indicates that when it comes to SCA no assumption about the behaviour of algorithms may be taken for granted and as a consequence of this designers need to verify the SCA resistance of their implementations thoroughly for each target frequency.

These results and additional details are published in [28], [105].

8 Vulnerability of atomic patterns

In this section we focus on the side-channel analysis attack resistance of the design supporting four different elliptic curves: two ECs over $GF(2^n)$ and two ECs over $GF(p)$, i.e. the NIST ECs *B-233*, *B-283*, *P-224* and *P-256*. The implementation of the ***kP*** calculation for the *P*-curves employs atomic patterns that are considered as a means to render simple side-channel analysis attacks void. But our investigation shows that the application of the atomic patterns introduced in [2] is not sufficient to prevent either simple SCA or horizontal address-bit DPA attacks, which are both single-trace attacks. Thus, not only the regular Montgomery ladder algorithm investigated in previous sections but also the *double-and-add* algorithm using atomic patterns [2] are vulnerable to horizontal SCA attacks, even to simple SCA. The SCA leakage is caused by the key-dependent addressing of blocks/registers, in both algorithms. Thus, the assumption about the indistinguishability of the addressing of the design blocks/registers has to be revised, at least for hardware implementations of ***kP*** algorithms.

In this section we discuss the vulnerability of our implementation of the ***kP*** design corresponding to the atomic patterns proposed in [2] and show that the strategy described in the previous section, i.e. using the hiding capability of the field multiplier, can reduce the success of horizontal attacks significantly. Designers have to be aware of such details and avoid a straightforward implementation of cryptographic algorithms as that may result in a design vulnerable even to simple SCA attacks.

8.1 Implementation Details

For our implementation of ***kP*** operations we selected an algorithm that is fast and, in the literature, mentioned as resistant against simple SCA, due to the atomicity principles of the algorithm. This is the atomic patterns algorithm using mixed Jacobian-affine coordinates [2] for ECs over *GF(p)*. The resistance of the atomic patterns algorithm [2] against horizontal address-bit DPA was, to the best of our knowledge, not evaluated yet. We assumed that the power profiles of point doublings and point additions, implemented corresponding to atomic patterns described in [2], are distinguishable due to the key-dependent use of registers in these patterns, which is due to the fact that the addressing of different registers is a distinguishable operation. Thus, algorithm [2] is – theoretically – vulnerable to horizontal address-bit DPA too, similar to regular Montgomery ladder using projective Lopez-Dahab coordinates which we used for implementing ***kP*** algorithms for ECs *B-233* and *B-283*. But we expected that the resistance of our design for 4 ECs is higher than the one of a design for a single EC due to the fact that some operations (for example field reductions) are always performed for all 4 ECs in our design. This increases the energy consumption of the ***kP*** design but it is a kind of a noise that can hide the key-dependent addressing of the registers. Our measurement and analysis results confirmed these assumptions.

We implemented elliptic curve point operations in hardware and selected the following algorithms for the implementation:

- the modified Montgomery ***kP*** algorithm using Lopez-Dahab projective coordinates as described in [75] for acceleration of the ***kP*** operations using the ECs *B-233* and *B-283, i.e.* we modified the IHP basic ***kP*** design (see details in sections 2.4 and 4) for performing operations with both ECs;
- the *double-and-add* ***kP*** algorithm with atomic patterns using mixed Jacobian-affine coordinates for point doublings and point additions (see patterns given in [2]) as the basis for our accelerator for the ECs *P-224* and *P-256*. We applied the Straus-Shamir

trick [56]-[57] as recommended in [2] for the calculation of $u\boldsymbol{P}+v\boldsymbol{Q}$. We slightly modified the sequence of operations given in [2] with the goal to increase pipelining of field operations.

Due to the fact, that we already discussed the $k\boldsymbol{P}$ implementation for *B*-curves, we concentrate here only on the implementation of the *P*-curves. A summary of implementation details for the *B*-curves is given in Appendix 1. More design's implementation details are given in [106], [134]-[136].

Our $k\boldsymbol{P}$ implementation is a bitwise processing of the scalar k. The Straus-Shamir trick (see Algorithm A1 in [106]) allows to perform only one $k\boldsymbol{P}$ operation instead of two independent elliptic curve point multiplications $u\boldsymbol{P}$ and $v\boldsymbol{Q}$. The number of point doublings is unchanged but the number of point additions is increased. This trick reduces the execution time of the ($u\boldsymbol{P}+v\boldsymbol{Q}$) calculation significantly.

Pipelining of the field operations in our implementation as well as the applied 4-segment Karatsuba multiplication formula for realizing the dual-field multiplier reduce the execution time of each $k\boldsymbol{P}$ operation by about 40% compared to the classical multiplication method. The time reduction has a similar effect as applying the Straus-Shamir trick.

The algorithm selected for the $k\boldsymbol{P}$ implementation as well as the 4-segment Karatsuba multiplication formula for the prime Galois fields can be found in [106] (see Algorithm A2 and Equation (A1) in [106]). In the rest of this section, we describe the implemented sequence of the field multiplications and explain the structure of our design and its blocks.

TABLE XXV. shows the operation sequence we implemented in our $k\boldsymbol{P}$ design for ECs over *GF(p)* based on the algorithm given in [2]. The processing of a key bit value k_i =0 requires only a single point doubling. The processing of a key bit value k_i =1 requires two point operations – a point doubling and a point addition. In the algorithm given in [2], a subtraction is performed after the first field multiplication and after these two operations an addition of the field elements is executed. In our implementation, we perform the addition before the subtraction, in parallel to the first multiplication. Operations performed in parallel are shown in the same row in TABLE XXV.

TABLE XXV. OPERATION FLOW FOR POINT DOUBLINGS AND POINT ADDITIONS IN OUR DESIGN FOR ECS OVER *GF(P)*

	point doubling: $2\boldsymbol{S}$	**point addition: $\boldsymbol{S}+\boldsymbol{P}$**
	inputs: $\boldsymbol{S} = (X_1 : X_2 : X_3 : Z_1 : Z_2)$	inputs: $\boldsymbol{S} = (X_1 : X_2 : X_3 : Z_1 : Z_2)$ $\boldsymbol{P} = (X : Y : Z_a)$ – input point
M1	$R_0 \leftarrow X_3 \cdot X_3$	$R_1 \leftarrow X \cdot Z_1$
*Add*1	$R_2 \leftarrow X_2 + X_2$	$\boldsymbol{R_2 \leftarrow X_2 + X_2}$
*Add*2	$R_1 \leftarrow X_1 - R_0$	$R_1 \leftarrow R_1 - X_1$
M2	$Z_1 \leftarrow X_2 \cdot R_2$	$R_2 \leftarrow R_1 \cdot R_1$
*Add*3	$X_2 \leftarrow Z_1 + Z_1$	$\boldsymbol{R_0 \leftarrow R_2 + R_2}$
M3	$R_3 \leftarrow R_2 \cdot X_3$	$R_3 \leftarrow X_1 \cdot R_2$
M4	$R_2 \leftarrow X_2 \cdot X_1$	$R_0 \leftarrow Y \cdot Z_2$
*Add*4	$X_1 \leftarrow X_1 + R_0$	$\boldsymbol{Z_2 \leftarrow Z_2 + R_0}$
M5	$R_0 \leftarrow R_1 \cdot X_1$	$Z_2 \leftarrow R_1 \cdot R_2$
M6	$R_1 \leftarrow Z_1 \cdot X_2$	$R_2 \leftarrow X_3 \cdot R_1$
*Add*5	$X_1 \leftarrow R_0 + R_0$	$X_1 \leftarrow R_3 + R_3$
*Add*6	$R_0 \leftarrow R_0 + X_1$	$X_1 \leftarrow Z_2 + X_1$
M7	$X_1 \leftarrow (R_0)^2$	$Z_1 \leftarrow (X_1)^2$
*Add*7	$X_1 \leftarrow X_1 - R_2$	$R_0 \leftarrow R_0 - X_2$
M8	$Z_1 \leftarrow (R_3)^2$	$R_1 \leftarrow (R_0)^2$
*Add*8	$X_1 \leftarrow X_1 - R_2$	$X_1 \leftarrow R_1 - X_1$
*Add*9	$R_2 \leftarrow R_2 - X_1$	$R_1 \leftarrow R_3 - X_1$
M9	$Z_2 \leftarrow Z_1 \cdot R_3$	$R_3 \leftarrow R_1 \cdot R_0$
M10	$X_2 \leftarrow R_0 \cdot R_2$	$R_0 \leftarrow X_2 \cdot Z_2$
	$X_3 \leftarrow R_3$	$X_3 \leftarrow R_2$
*Add*10	$X_2 \leftarrow X_2 - R_1$	$X_2 \leftarrow R_3 - R_0$
	output: $2\boldsymbol{S}=(X_1 : X_2 : X_3 : Z_1 : Z_2)$	**output:** $\boldsymbol{S}+\boldsymbol{P}=(X_1 : X_2 : X_3 : Z_1 : Z_2)$

The field operations in *GF(p)* require more gates when implemented in hardware and are slow compared to operations in $GF(2^n)$. This is due to the carry bit propagation and the more complex reduction. We implemented squaring operations as multiplications with the same operands. The sequence of operations in our design for point doublings as well as for point additions consists of 10 field multiplications, 10 additions (subtractions) and write to register operations. Thus, we realized the operation flow given in TABLE XXV. using a field multiplier block, a field adder and 12 registers: X_1, X_2, X_3, Z_1, Z_2; X, Y, Z_q and R_0, R_1, R_2, R_3. The registers X and Y contain the coordinates of the input point ***P*** and will be not written but only read during the point doublings as well as during the point additions.

The operations listed in TABLE XXV. , as well as operations required for the Montgomery ***kP*** design for ECs over binary extended fields, define the structure of our ***kP*** design. It consists of:

- a dual-field multiplier, denoted further as **MULT**, that can calculate a field product for each of the 4 selected Ecs, i.e. *B-233*, *B-283*, *P-224* and *P-256*;
- an **ALU** block for addition/subtraction in both *GF(p)* fields as well as addition and squaring operations in both $GF(2^n)$ fields;
- **Registers** for storing of the inputs, outputs and intermediate data;
- **A Controller** that manages the data flow between the design components and defines which operation has to be performed in the current clock cycle, including storing the data into registers as well as reading the data from the registers, ALU and multiplier;
- A **BUS** that implements the data exchange between the blocks and the registers corresponding to the Controller signals.

The structure of our design is illustrated in Fig. 69.

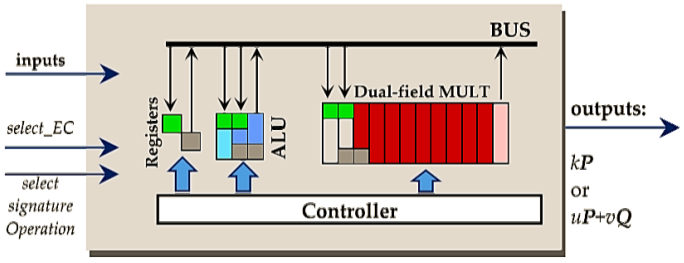


Fig. 69. Structure of the design for ***kP*** operations for 4 ECs.

The BUS t is implemented as a multiplexer. It consists of many logic gates that react on the address given by the Controller. Only one of the blocks can write its output data to the BUS in a certain clock cycle, but in principle all blocks could read data from the bus at the same time. The *write-to-BUS* operation from the Controller connects the output of the source block to the inputs of all other blocks. By the *read-from-BUS* operation, only the destination block accepts the values on its input as data for processing.

Reading data from the BUS and storing the data into a register requires two clock cycles to be performed:

- In the first clock cycle, the data are written to the BUS, whereby the selected register (or the other selected design block) obtains the signal we (word enable) from the Controller. A green rectangle in Fig. 69 indicates the addressing of the Registers, ALU and MULT blocks;
- In the second clock cycle, the data are stored into the previously addressed register or into an internal register of the previously addressed block, data storing is denoted with a small grey rectangle in Fig. 69.

The block ALU can perform different operations depending on the Controller signals as follows:

- reading data from the BUS (green rectangle); the data will be stored into an internal register (grey rectangle in the block ALU);
- squaring of $GF(2^n)$-elements (marked by yellow rectangles)
- addition of field elements (marked by dark blue rectangles) in prime or binary extended fields.
- subtraction of $GF(p)$-elements (marked by magenta rectangles)
- writing data to the BUS.

The block MULT performs a field multiplication for one of the four ECs. Depending on the signals from the Controller, the MULT block:

- obtains two multiplicands: reading data from the BUS (see two green rectangles) and storing the operand values into internal registers (see two grey rectangles in block MULT). This requires two clock cycles;
- calculates partial products according to the 4-segment Karatsuba multiplication formula, accumulates the partial products in its internal register and performs a field reduction. In each clock cycle a single partial product will be calculated. Corresponding clock cycles are marked with a red rectangle. The reduction of the intermediate product value is also performed in each clock cycle, after the accumulation of the currently calculated partial product. This increases the energy consumption of the field multiplier slightly but also hides partially the key-dependent activity/addressing of other design blocks. Due to the applied 4-segment Karatsuba multiplication formula 9 different partial products have to be calculated, i.e. the field multiplier needs 9 clock cycles for the accumulation of a field product;
- writes the calculated field product to the BUS directly after the 9 partial products are accumulated;
- stalls, i.e. calculates a partial product of two zero operands. During the calculation of the first such partial product directly after the processing of non-zero operands, the multiplier consumes a slightly reduced energy (this clock cycle is marked by light red). The energy consumption of the multiplier is small while stalling. We marked such clock cycles by white colour.

Taking into account the results of investigations described in section 6.1, we implemented the partial multiplier in our MULT block using the classical multiplication formula. It also allowed us to reach the maximal operation frequency and simplify the implementation for both finite fields.

Fig. 70 shows the operation sequence for a point doubling and a point addition in our implementation for ECs over $GF(p)$ providing details about activities of each block.

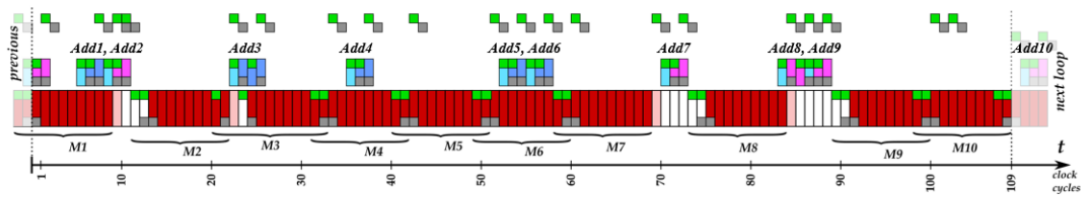


Fig. 70. Operation sequence for a point doubling or a point addition in our implementation for ECs over $GF(p)$, corresponding to TABLE XXV. .

For point additions using mixed Jacobian-affine coordinates the atomic patterns proposed in [2] do not allow such a high level of parallelization and pipelining. Even in our, slightly

modified sequence of operations, the addition *Add3* and the subtractions *Add2*, *Add7*, *Add8* and *Add9* can be performed only between multiplications. Due to the realized pipelining and the fast field multiplier a single atomic pattern – a point doubling or a point addition – requires only 109 clock cycles. Thus, a key bit value '0', that requires only a point doubling, can be processed in 109 clock cycles. The processing of a key bit value '1' requires two atomic patterns (a point doubling and a point addition), i.e. it requires 2·109=218 clock cycles. Please note, that a design that uses the classical multiplication method instead of the 4-segment Karatsuba multiplication formula would require 179 clock cycles for a single atomic pattern. Thus, the use of the 4-segment Karatsuba multiplication formula reduces the execution time of a single atomic pattern by about 40%[25], whereby the maximal frequency, determined by the longest signal propagation path of the partial multiplier and the reduction unit, is the same.

8.2 Horizontal DPA attack

We synthesized our ECDSA accelerator for the IHP 250 nm technology and generated power traces of a $k\boldsymbol{P}$ execution for each of the 4 ECs (more details about the trace simulations are given in section 3.1.1). The maximum achieved clock frequency is 62.5 MHz (16 ns clock cycle period) after synthesis. Power traces were simulated with a step of 0.01 ns, i.e. our trace contains 1600 simulated values per clock cycle[26]. To simplify the analysis and due to the fact, that the simulated trace is noiseless, we performed a trace compression. For each clock cycle we calculated the sum of its sampling values and represented it using this single value. (i.e. the *y*-axis shows the average power consumption in W, multiplied by 1600).
Fig. 71 shows parts of the compressed power traces simulated for the ECs *P-224* and *P-256*. The shown parts correspond to the processing of the two key bits '01', i.e. here three atomic patterns are shown: two point doublings, and a point addition.

The clock cycles with reduced energy consumption are easy to see in Fig. 71. As described above, the non-active field multiplier is the reason for reduced energy consumption. These clock cycles are a kind of marker. They "show" the periodicity of an atomic pattern. The knowledge about the period of atomic patterns is helpful for the preparation of different attacks. In contrast to the atomic patterns for the point operations in *GF(p)*, the atomic patterns corresponding to the processing of a key bit in $GF(2^n)$ do not have such a kind of markers in our implementation.

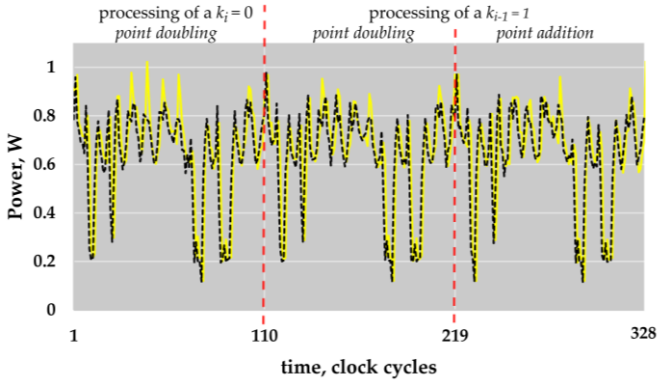


Fig. 71. Part of the simulated power traces of a $k\boldsymbol{P}$ execution corresponding to the processing of two key bits '01' for ECs *P-224* (black dotted line) and *P-256* (yellow line):.

[25] $40\% \approx \frac{179-109}{179} \cdot 100\%$

[26] We do not need such a fine-grained simulation for experiments described in this work. We did it with the goal to be able to use the traces in future investigation too.

Due to the duration of an atomic pattern (in clock cycles) and the compression of the power trace (only a single value represents each clock cycle), the power profile of an atomic pattern consists of 109 sample values for ECs over *GF(p)*. We denote the current sample in an atomic pattern power profile (further atomic profile) using letter *i* in this work ($i \in \{1,2, ..., 109\}$). We numbered each power profile of the processing of a key bit k_j in the main loop of the implemented algorithm using the same index letter *j*. Please note that in the main loop in our implementation for ECs over *GF(p)* $j \in \{l-2, l-1, ..., 1,0\}$.

We analysed the ***kP*** traces for ECs *P-224* and *P-256* in a similar way as described in section 3.2.3. The difference is only that we tried to distinguish 2 kinds of atomic patterns, EC point additions and EC point doublings. We calculated the mean power profile of point addition/doubling operations performed in the main loop of the ***kP*** executions for ECs *P-224* and *P-256*. The mean profile contains 109 samples – one sample per clock cycle. We compared the mean profile of the point operation calculated for a ***kP*** execution with each point operation profile in the ***kP*** execution trace sample-wise. Using the mean atomic profile, we made an assumption about the performed EC point operations, i.e. we distinguished point additions and point doublings. We denoted each point doubling profile followed by a point addition profile as processing of a key bit value '1', corresponding to the implemented *double-and-add* algorithm. For the evaluation of each extracted key candidate, we calculated its correctness as described in section 3.2.1.

Thus, for each ***kP*** trace we extracted 109 key candidates. Fig. 72 shows the attack results for the ECs *P-224* and *P-256*.

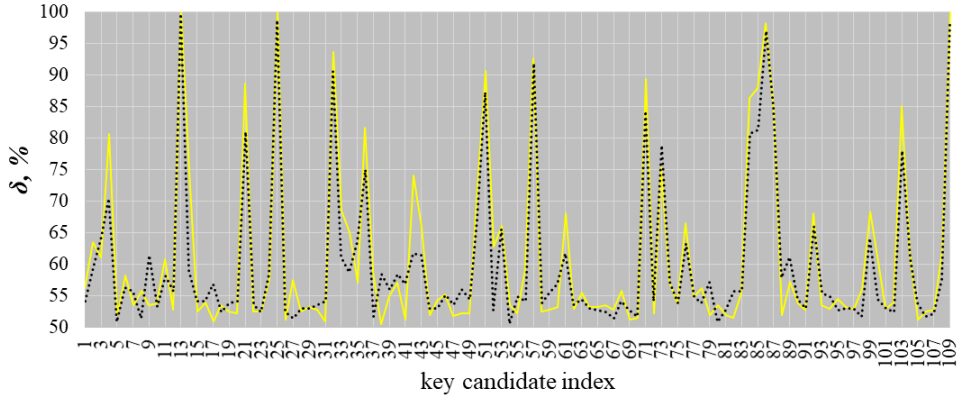


Fig. 72. Attack results: ECs *P-224* (black dotted line) and *P-256* (yellow line).

The results of the performed attacks demonstrate that the atomicity patterns do not prevent horizontal attacks from being successful. Attacking EC *P-224* we obtained 4 key candidates with a correctness higher than 97% (see key candidate 13, 25, 86 and 109). Attacking EC *P-256* we fully revealed the key (see key candidates 13, 25 and 109 with a correctness of 100%).

The weak point of the atomicity principle is the assumption that the addressing of different registers (or other blocks of the design) is indistinguishable[27]. This does not hold true if the design is a hardware implementation.

We ported our design to an Arty Z7-20 FPGA with the goal to confirm the conducted investigations on a real device (the measurement setup is described in section 7.2). The maximum operating frequency for the design is slightly above 25 MHz. However, we applied a clock frequency of 10 MHz to increase the number of captured samples per clock cycle during the collection of the ***kP*** traces.

[27] The same weak point have cryptographic designs implementing the regularity principle as a countermeasure against simple SCA attacks.

We repeated the *comparison to the mean* attack against the measured traces of the ***kP*** execution. All the design inputs for the ***kP*** operations, i.e. coordinates of the EC points as well as scalars for each of the curves, were kept the same as those used during the simulations of the power traces. Fig. 73 shows the attack results for the ECs *P-224* and *P-256*. Attacking EC *P-224* we obtained 4 key candidates with the correctness of 100%, i.e. we fully revealed the key (see key candidates 13, 73, 86 and 87). Attacking EC *P-256* we obtained 5 key candidates (clock cycles 13, 73, 86, 87 and 89) that are identical to the processed key.

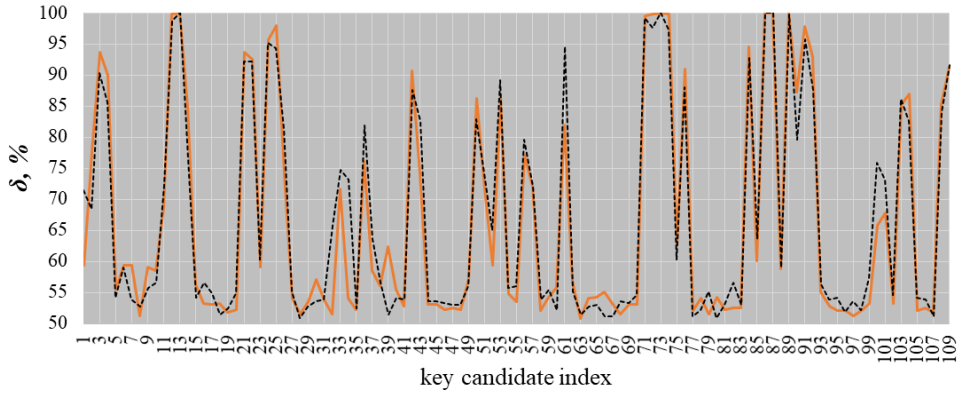


Fig. 73. Attack results for the EM traces measured on an FPGA: ECs *P-224* (black dotted line) and *P-256* (orange line).

The results of the attacks analysing simulated and measured traces show that despite the significant difference between the target platforms (i.e. ASIC and FPGA), the results are quite similar, i.e. we were able to reveal the scalar used by ***kP*** execution completely. The number of the key candidates revealed with a correctness of 100% attacking FPGA traces is even higher compared to those obtained by attacking simulated traces of ASICs. However, these key candidates point to the same SCA leakage source within the slot for both target platforms.

8.3 Automated simple analysis attack

We decided to investigate our design additionally analysing the uncompressed measured electromagnetic traces of the elliptic curves *P-224* and *P-256*. Analysis of uncompressed traces can give advantages for designers for determining the leakage sources as well as for attackers when revealing the key. If the leakage source is not very high/strong, or if the leakage duration is short in comparison to the one of a clock cycle, the compression can "hide" this leakage, at least partially.

The main purpose of the atomicity principle is to protect elliptic curve scalar multiplications against simple side-channel analysis attacks. Therefore, we applied an automated simple analysis attack (see section 3.2.5.1) against the measured traces, i.e. we tried to distinguish the point doubling atomic pattern from the one of the point addition using our SCA Toolkit.

As already mentioned before, the design was running at a 10 MHz clock frequency and the traces were captured by an oscilloscope at a 10 GS/s sampling rate. These parameters result in 1000 measured samples per clock cycle. Hence, a single atomic pattern, consisting of 109 clock cycles in our implementation, is represented by 109000 values within the measured trace.

In total, for each of the traces, we identified 994 samples out of 109000 samples in the atomic pattern, for EC *P-224* and 1122 samples for EC *P-256* that allow to successfully reveal the processed scalar using simple visual inspection. Detailed information about the distribution of such samples within the atomic pattern as well as visualisation of such data are given in TABLE XXVI. and Fig. 74.

TABLE XXVI. DISTRIBUTION OF THE "LEAKY" SAMPLES IN THE ATOMIC PATTERN

Clock cycle	12	13	24	71	72	73	74	86	87	89	91
	Number of the samples with a high leakage										
EC *P-224*	-	470	57	48	-	205	-	131	72	11	-
EC *P-256*	4	503	19	65	87	200	2	99	116	24	3

Taking into account the information given in Fig. 70 and Fig. 74, it is obvious that all the significant leakage comes from the clock cycles of the atomic pattern in which the multiplier block is not active. Multiplier's inactivity results in a significant reduction of the energy consumed. Thus, the "contribution" of the other blocks to the energy consumption of the whole design becomes more "visible".

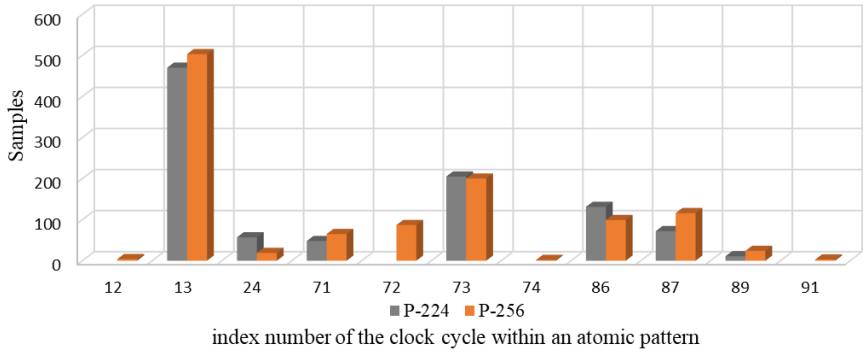


Fig. 74. Visualization of the data from TABLE XXVI. Grey and orange bars correspond to the data obtained analysing a ***kP*** trace for the ECs *P-224* and *P-256* respectively.

We use an example of the electromagnetic trace for the EC *P-256* to visualize the difference between the shapes of the point doubling and the point addition atomic patterns[28], indicated by I – IV in Fig. 75-*a*. Parts I-IV are also shown in Fig. 75-*b* – Fig. 75-*e* zoomed-in. The shapes of point additions and point doublings are represented by magenta and green areas respectively.

As it can be seen in Fig. 75, the four regions with an inactive multiplier differ significantly in the level of the SCA leakage. Please note that the figures have different *y*-axis scales. The most "visible" leakage is located in the 13th clock cycle of the atomic pattern (see Fig. 75-*b*). The green and magenta shapes are completely separated from each other in most of the points for the *x*-axis values in the range between 12000 to 13000. The leakage in clock cycle 24 is almost "invisible" as there is only a small separation close to the sample number 23320 see Fig. 75-*c*. From the designers' point of view, the best case in terms of resistance against simple analysis attacks corresponds to completely grey areas, i.e. the overlapping ones, where the operations' execution profiles cannot be distinguished from each other.

The multiplier is a big block of our ***kP*** design. It consumes all the DSP blocks, almost a half of the LUTs (49,94%) and 21,92% of flip-flops required by the whole design (for more details about the utilized resources see [106]). Its energy consumption is high in comparison to other design blocks. Thus, the activity of the multiplier is a kind of noise that can hide the activity of other blocks. In the next section we demonstrate how the multiplier's activity can reduce the success of horizontal attacks and simple analysis attacks as particular case.

[28] Visualization for the EC *P-224* looks similar and is given in [106].

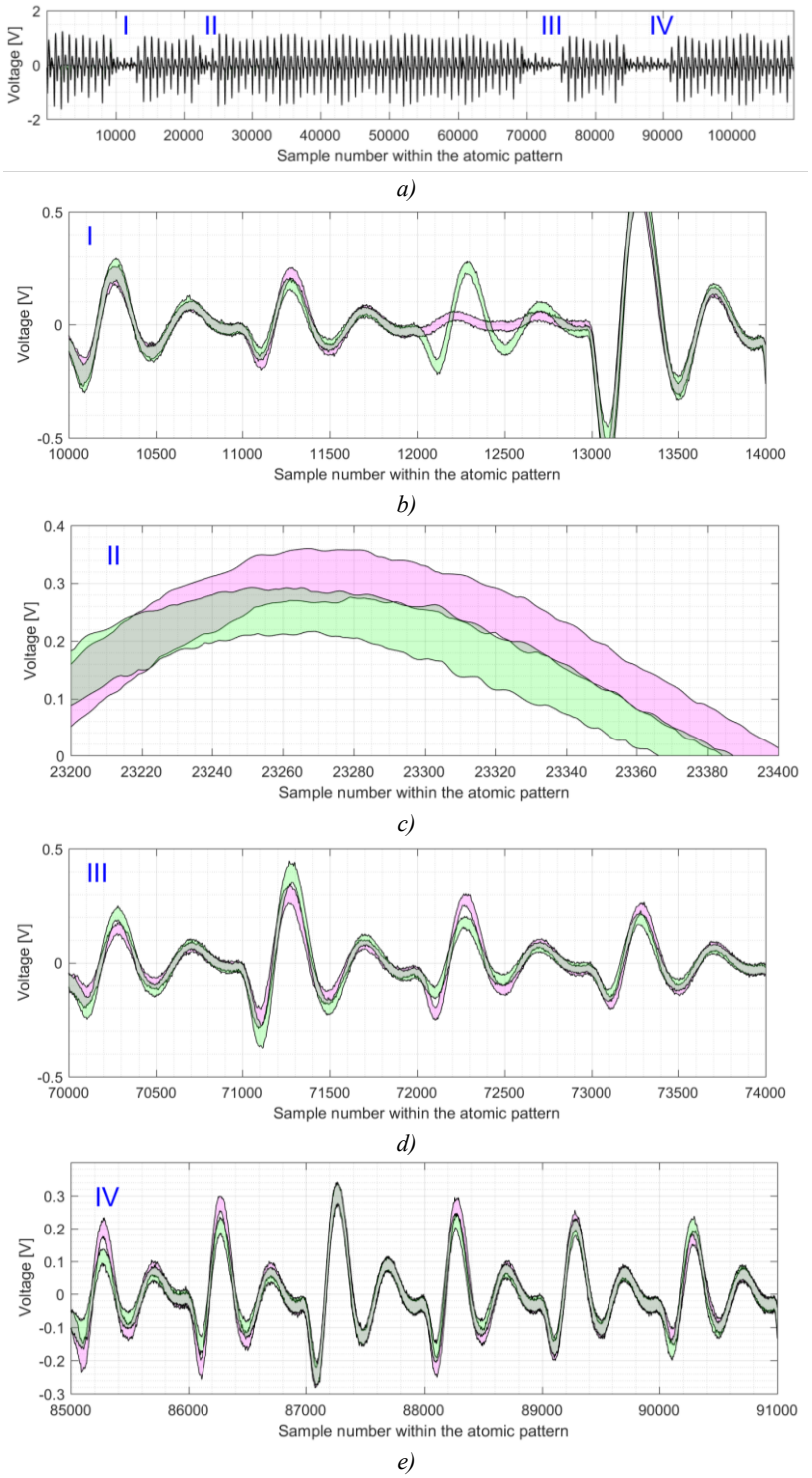


Fig. 75. Shapes for the processing of EC point doublings (see green marked area) and EC point additions (see magenta marked area) for the *P-256* elliptic curve: (a) shows the shapes of all atomic patterns with 4 regions (zoomed-in on (b)-(e) respectively) where the multiplier is not active; (b) shows the region that corresponds to clock cycles 11 to 14; (c) shows a part of the 24th clock cycle; (d) shows clock cycles 71 to 74 of the atomic pattern; (e) shows clock cycles 86 to 91 of the atomic pattern.

8.4 Dummy partial multiplications as a countermeasure

We modified our unified design by adding execution of dummy partial multiplications in clock cycles where the multiplier block was not active before. The design was ported to the same FPGA. In comparison to the original design the modification with dummy partial multiplications introduced an overhead of only about 1,3% in terms of LUTs utilisation. The most affected block is the multiplier which was increased insignificantly.

We measured electromagnetic ***kP*** traces on the modified design using the same inputs as in the previous experiments and analysed them using the comparison to the mean approach. The attack results for the *P-224* and *P*-256 elliptic curves are presented in Fig. 76 and compared with the original design, i.e. the design without the dummy activity of the field multiplier.

As it can be seen in Fig. 76, the correctness of key candidates for most of the clock cycles in the atomic pattern regions where the multiplier was not active, i.e. clock cycles 11 to 13, 24, 71 to 75 and 86 to 91 (see black dotted and orange lines) significantly dropped down (see violet and brown lines).

The number of key candidates for the EC *P-224* revealed with a correctness of more than 90% dropped down from 18 to 10 in comparison to the original design while the best key candidate has a correctness of 97,49%. For the EC *P-256* the number of key candidates with a correctness of more than 90% dropped down from 20 to 9. The best key candidate has a correctness of 95.78%. Thus, the application of dummy partial multiplications reduces the success of the comparison to the mean attack.

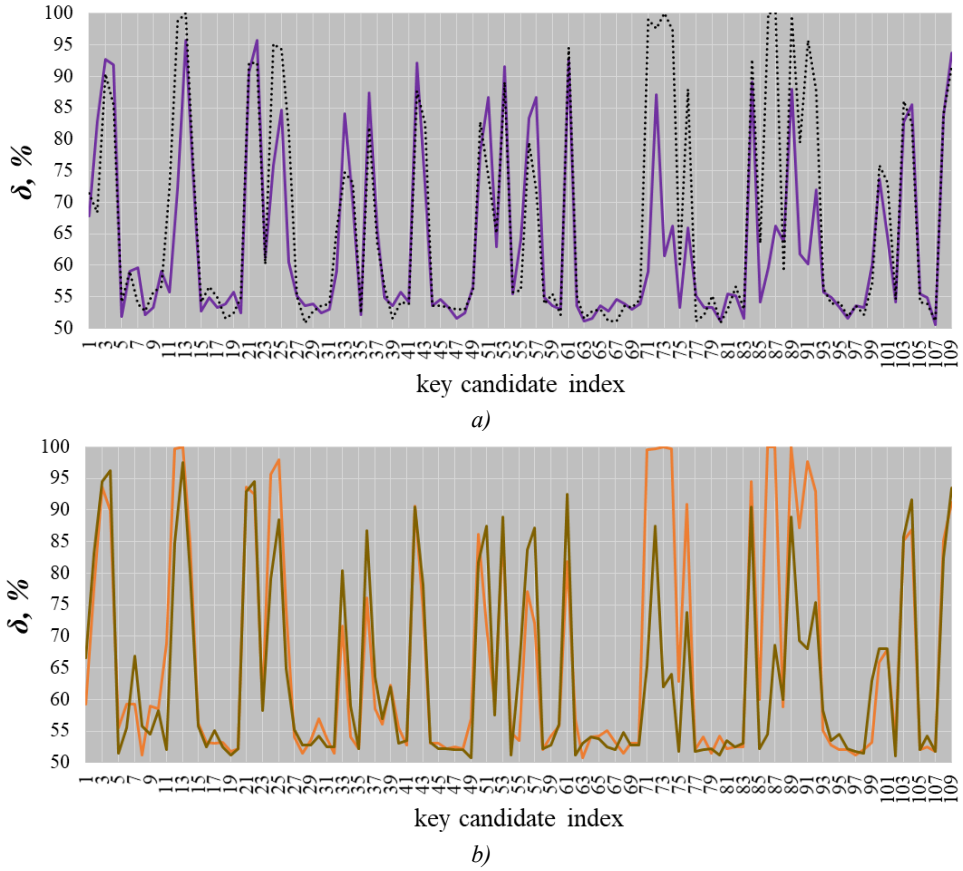


Fig. 76. Attack results for the ***kP*** traces of ECs *P-224* (a) and *P-256* (b). Violet and brown lines correspond to the design with dummy partial multiplications; black dotted and orange lines correspond to the original design and are given as a reference.

Additionally, we repeated the simple analysis attack against the design with dummy partial multiplications using traces of the *P-224* and *P-256* ***kP*** executions. As a result, we discovered only 17 out of 109000 samples that allow revealing the full key for the *P-224* trace. All "leaky" samples are located in clock cycle 13 of the atomic pattern.

The number of samples, that allow revealing the full key decreased from 994 down to 17 samples for EC *P*-224 and from 1122 to 0 samples for EC *P-256*.

The differences in shapes of the point doublings and point additions are demonstrated in Fig. 77-*a* and -*b* for the clock cycle 13 of the atomic pattern for *P-224* and *P*-256, respectively.

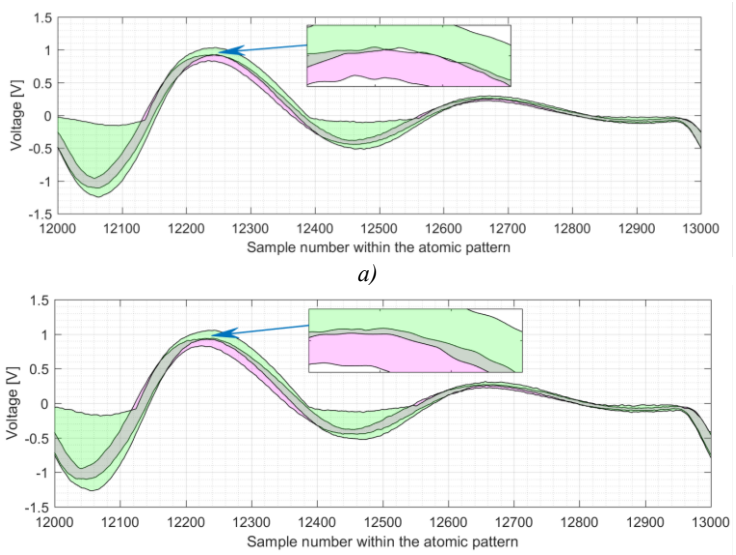


Fig. 77. Shapes for the processing of EC point doublings (see green marked area) and EC point additions (see magenta marked area) in clock cycle 13 for *P-224* (a) and *P-256* (b).

In this section we showed that the application of the atomic patterns introduced in [2] is not even sufficient to prevent simple SCA attacks. The source of the leakage is related to the key-dependent addressing of design blocks/registers in the atomic patterns, i.e. the assumption about the indistinguishability of the addressing of the design blocks/registers has to be revised. The noise produced by the activity of the field multiplier can reduce the success of applied attacks and therefore increase the inherent resistance of the whole design. We published our investigations about the vulnerability of atomic patterns to horizontal attacks in [106], [137]-[139].

9 Conclusion

In this work we concentrated on the EC point multiplication with a scalar, denoted as ***kP***. The ***kP*** operation is the main operation of EC cryptographic protocols. The observation of so-called side-channel effects – power consumption, electromagnetic radiation, execution time, etc. – during a single execution of the ***kP*** operation can be exploited for successfully revealing of the scalar *k*. This is the goal of many attacks against cryptographic protocols. We investigated the resistance of different ***kP*** implementations of Montgomery ladder for EC over $GF(2^n)$ against single-trace SCA attacks. The Montgomery ladder is a highly regular binary ***kP*** algorithm, i.e. the scalar *k* is processed in the main loop of the algorithm bit-by-bit, whereby the sequence of the operations for the processing of each bit of the scalar does not depend on the processed bit's value. The key-dependent differences are only the registers for storing or reading the data. Due to the regularity of the Montgomery ladder and the assumption that operations with different registers are not distinguishable, the Montgomery ladder is known in the literature as resistant against simple SCA attacks. The success of vertical address-bit DPA attacks [44] against a hardware implementation of the Montgomery ladder demonstrated that the addressing of registers can be exploited for revealing the key. In this work we were able to reveal the key exploiting the distinguishability of addressing of the registers in single trace attacks. The key-dependent addressing of the registers or other design blocks is an intrinsic feature, not only of the Montgomery ladder, but also of binary ***kP*** algorithms based on the atomicity principle. In this work, we implemented a ***kP*** accelerator for ECs over *GF(p)*, i.e. for the standardized ECs *P-224* and *P-256* corresponding to an atomic pattern algorithm [2]. Analysing single ***kP*** execution traces statistically as well as performing automated simple SCA we were able to reveal the processed scalar *k* successfully. We demonstrated that the reason causing the successful revealing of the key is the key-dependent addressing of the design blocks and registers in both cases, i.e. not only in the case of the Montgomery ladder but also in the case of atomic pattern algorithms. The success of our attacks shows that the regularity and atomicity principles are not effective countermeasures against horizontal address-bit SCA attacks. We proposed to use the hiding effect of the field multiplier as well as regular scheduling for adressing the blocks as additional means to increase the resistance of the ***kP*** designs. Applying these means in regular and/or atomic pattern algorithms can increase the resistance of the designs significantly. Some ideas for increasing the resistance of ***kP*** implementations against address-bit SCA attacks were registered as 2 patents. In the following we summarize the main contributions of this thesis in more detail:

1. We determined the SCA leakage sources and their nature:

   The success of horizontal attacks is caused by the key-dependent addressing of the design blocks.

   Corresponding to the state-of-the-art literature, the influence of the key-dependent addressing on the resistance of the design is explained with a different hamming distance of the addresses of the previously and currently addressed blocks. Our experiments showed that the easy solution – to make the hamming distance between addresses of each pair blocks – is possible but it is not effective. This is due to the fact, that not only the hamming distance of the block's addresses influences the energy consumption and – consequently – the resistance of the design, but also the type of the addressing, i.e. for writing or for reading the data to/from the data bus, as well as implementation details of the addressing on the hardware level.

   The main implication is that many regular algorithms such as the Montgomery ladder or algorithms based on atomicity principles are vulnerable to horizontal (single-trace) address-bit SCA attacks.

2. We investigated the influence of state-of-the-art countermeasures on the resistance of the design.

   Well-known countermeasures were aiming to dissolve the dependency between the processed scalar *k* and the coordinates of the EC point *P*. These countermeasures are effective against the so-called data-bit vulnerability, but not against the address-bit vulnerability. Horizontal attacks are especially dangerous, due to the fact that countermeasures that are effective against vertical attacks, are not effective against horizontal attacks. This holds true even for countermeasures aiming to prevent address-bit attacks. We investigated some countermeasures such as randomization of the execution order of the operations in the main loop of the ***kP*** algorithm as well as GALS-ification against horizontal address-bit attacks. In our experiments we were able to reveal the processed scalar completely despite these countermeasures were implemented.

3. We proposed to use the activity of the field multiplier for – at least partially – hiding the SCA leakage sources

   Based on the results of our investigations, we proposed to use the activity of the field multiplier that is the biggest block in the design as a kind of countermeasure. This approach is able to reduce the success of many SCA attacks – horizontal and vertical, data-bit and address-bit. We evaluated our idea on the example of ***kP*** designs for 4 different ECs. We experimented with the IHP ***kP*** designs for binary ECs, i.e. *B-233* and *B-283* and we implemented and evaluated the designs for ECs over prime fields, i.e. *P-224* and *P-256*. Our experiments show that the activity of the field multiplier implemented corresponding to the 4-segment Karatsuba multiplication formula, in conjunction with the classical multiplication formula for the partial multiplier, can improve the resistance of the design significantly. But this as a single means is not sufficient, i.e. it has to be combined with other kinds of countermeasures.

4. To the best of our knowledge, it is the first implementation of a ***kP*** accelerator for EC over prime finite fields *GF(p)* exploiting the advantages of the 4-segment Karatsuba multiplication method. Its application leads to the fact that the execution time and – consequently – the energy consumed by the ***kP*** operation was reduced by about 40 %.

5. We proposed regular scheduling for the addressing of the blocks of the design.

   This approach is an effective strategy for reducing the success of horizontal attacks in all clock cycles in which it is applicable. Combining this approach with hiding features of the field multipliers can increase the resistance of the design significantly.

6. We showed that "easy" and low-cost analysis methods such as the *comparison to the mean*, an automated simple SCA analysis and statistical analysis in the frequency domain can be very effective means for attackers. Designers can benefit from these approaches as well since they can be successfully applied to determine SCA leakage sources in early design phases.

7. We demonstrated on the example of our internally-developed cryptographic core, how the security strength of the design depends on the applied target platform as well as semiconductor manufacturing process and the logic gate libraries. The same cryptographic design, implemented in VHDL and ported to different FPGAs, may evidence different resistance to SCA attacks [140]. The same holds true even when the design was synthesized using different compiling options, for different target frequencies on the same target platform [28], [140] as well as for different technology nodes [141]. As a result, the cryptographic core designed and evaluated for a specific technology process and operating

conditions will require a new evaluation to confirm the resistance to potential SCA attacks in case any of the original conditions were changed.

8. Last but not least, a recommendation for the implementation of a design performing operations according to the ECDSA algorithm should be mentioned.

For such a design, *kP* operations involved in the signature generation and signature verification operations have to be implemented using different algorithms. On one side this will increase the area of the design. On the other hand, the verification in such a case can be implemented according to a fast and not secure algorithm which will increase the possible number of verifications per second (a critical parameter in automotive applications). But most importantly, such an approach can prevent the possibility of learning weak points of the *kP* implementation and preparing different template attacks against the signature generation, i.e. this approach can eliminate the possibility of performing some SCA attacks.

9.1 Future work

Implementation and re-designing of cryptographic implementations are challenging and tedious tasks. Evaluation of the design's resistance against (selected) attacks and the certification process are complex, time- and resource-consuming tasks.

Many facts influencing design resistance and attack success were discussed in this thesis. This means that designers have to be attackers with a high level of expertise in many kinds of attacks. The quality of the cryptographic implementations and their resistance against a broad spectrum of the attacks depend directly on the knowledge of the designers. Thus, the subjective factor of expertise plays a highly important role when designing cryptographic chips resistant to physical attacks. Nowadays, designers need a set of rules and tools for the evaluation of the SCA resistance of designs under development in early stages. This means a design methodology for engineering of cryptographic chips resistant against a broad spectrum of physical attacks needs to be researched. We propose this as a next step, based on our findings presented in this thesis. It should be noted that we also plan to investigate the application of machine learning (ML) for the automation of chip designing. Cryptographic implementations are a well suited example to develop automated AI-based design methodology. This is due to the fact that for cryptographic hardware accelerators two different types of attacks i.e. SCA and FI attacks are known. Improving the resistance against FI may increase the SCA vulnerability and vice versa. So, here exists the issue of solving the balancing act between reliability – more fault tolerance – and higher SCA resistance.

Designing cryptographic ASICs is a highly sophisticated complex task which needs to take into account non-functional requirements such as energy consumption, area and last but not least resistance against physical attacks. While most designers are familiar with the former requirements the latter is in many cases only partially understood. Even worse it depends on the target manufacturing technology, frequency, placement of the logic gates in the chip, length of wires in the ASIC, tools and algorithms applied for layout generation, etc. So, to significantly improve the resilience of cryptographic designs AI methods can be integrated in the design flow. In such a case, a ML-based toolbox for existing VLSI electronic design automation tools can be developed to facilitate the design of cryptographic chips resistant against different physical attacks. An AI-based methodology for developing highly resistant cryptographic designs can be very helpful for the engineering of resilient systems.

Currently RRAM-crossbar structures are promising means for implementing artificial intelligence (AI) algorithms in hardware. Therefore, the resistance of AI-accelerators against physical attacks is as important as the one of crypto-accelerators. The resistance of RRAM-

based structures against different physical attacks, partially against SCA and manipulation attacks, is of general interest. To the best of our knowledge, the resistance of RRAM structures against SCA attacks was not investigated, yet. It can be a new broad research topic in the future.

Additionally, successful EMA analysis attacks in our work show the necessity of the investigations to develop countermeasures against them. We demonstrated that EMA attacks measuring the traces over a power supply capacitors are efficient non-invasive kind of attacks. Please note that each capacitor is – potentially – a perfect measurement point. They are especially dangerous from the SCA perspective due to the fact that the boards can be modified relatively easy: some capacitors can be added, removed or replaced by an attacker. Opposite to the localized EMA attacks, where different kinds of shielding can significantly reduce the success of the attacks, the capacitors are not shielded, which makes them Achilles' heel allowing easy collection of side-channel data.

This opens a potential investigation area aiming at preventing measurements of the electromagnetic field over the power supply capacitors of a cryptographic chip by integrating them into the chip. For example in [142], it was proposed to use the two top metal layers of ICs to realize an individual power supply for every single gate. The top GND layer and the next VDD layer, implemented as metal planes and as a broad strip, build a plane capacitor. This strategy may help to avoid many attacks and additionally make the supply voltage more stable. Replacing all external capacitors with this or a similar approaches would increase the robustness of cryptographic chips against semi-invasive or invasive attacks. These attacks require decapsulation of the chip as a preparation step. In case of integrated power supply capacitors, the integrated capacitor will be damaged during decapsulation and not work properly.

The advent of large-scale quantum computers in the coming years is expected to affect significantly the security of cryptosystems currently considered secure. According to the NIST, it took almost twenty years to deploy our modern public key cryptography infrastructure. Therefore, the development of quantum-resistant public-key cryptographic algorithms has started nowadays in order to be prepared for the day when a large-scale quantum computer capable to break many of the currently existing public-key cryptosystems will be built. Nevertheless, SCA attacks will remain relevant even for quantum-resistant algorithms. As was shown this year (2022), one of the four candidates of the 4th round for the Post-Quantum Cryptography competition was broken within several hours [143]. So, future algorithms have to be resistant not only to the attacks performed using quantum computers but also to traditional SCA attacks. As the hardware where such algorithms are executed is based on the CMOS technology, analysis methods proposed in this work can be relevant for the post-quantum cryptographic algorithms as well.

References

[1] D. Hankerson, A. J. Menezes, and S. Vanstone, Guide to Elliptic Curve Cryptography. Berlin, Heidelberg: Springer-Verlag, 2003.

[2] F. Rondepierre, "Revisiting Atomic Patterns for Scalar Multiplications on Elliptic Curves," in Smart Card Research and Advanced Applications, Nov. 2013, pp. 171–186. doi: 10.1007/978-3-319-08302-5_12.

[3] N. Koblitz, "Elliptic curve cryptosystems," Math. Comp., vol. 48, no. 177, pp. 203–209, 1987, doi: 10.1090/S0025-5718-1987-0866109-5.

[4] V. S. Miller, "Use of Elliptic Curves in Cryptography," in Advances in Cryptology — CRYPTO '85 Proceedings, Berlin, Heidelberg, 1986, pp. 417–426. doi: 10.1007/3-540-39799-X_31.

[5] J. López and R. Dahab, "Fast Multiplication on Elliptic Curves Over GF(2^m) without precomputation," in Cryptographic Hardware and Embedded Systems, vol. 1717, Ç. K. Koç and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 316–327. doi: 10.1007/3-540-48059-5_27.

[6] D. Hankerson, J. Lopez, and A. Menezes, "Software Implementation of Elliptic Curve Cryptography over Binary Fields," in Cryptographic Hardware and Embedded Systems — CHES 2000, Aug. 2000, pp. 1–24. doi: 10.1007/3-540-44499-8_1.

[7] T. Oliveira, J. López, and F. Rodríguez-Henríquez, "The Montgomery ladder on binary elliptic curves," Journal of Cryptographic Engineering, Apr. 2017, doi: 10.1007/s13389-017-0163-8.

[8] Information Technology Laboratory, "Digital Signature Standard (DSS)," National Institute of Standards and Technology, NIST FIPS 186-4, Jul. 2013. doi: 10.6028/NIST.FIPS.186-4.

[9] E. Barker, "Recommendation for Key Management Part 1: General," National Institute of Standards and Technology, NIST SP 800-57pt1r4, Jan. 2016. doi: 10.6028/NIST.SP.800-57pt1r4.

[10] NXP A1006: Secure Authenticator IC https://www.nxp.com/products/identification-and-security/authentication/secure-authenticator-ic-embedded-security-platform:A1006. (last accessed 25.02.2019).

[11] Infineon OPTIGA™ Trust B SLE95250 Product Brief https://www.infineon.com/dgdl/Infineon-OPTIGA_Trust_B_SLE95250-PB-v01_00-EN.pdf?fileId=5546d4625b04ae11015b0f3f7f1c332e (last accessed 01.11.2022).

[12] Atmel/Microchip CryptoAuthentication Family of Crypto Elements with Hardware-Based Key Storage https://www.microchip.com/content/dam/mchp/documents/OTH/ProductDocuments/Brochures/Atmel-8756-ATSHA204A-ATAES132A-ATECC108A-ATECC508A-Flyer-E.pdf. (last accessed 01.11.2022).

[13] NXP Application note AN11875 A1006 Host Reference Implementation for LPC1115 (2016)

[14] D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *IJIS*, vol. 1, no. 1, pp. 36–63, Aug. 2001, doi: 10.1007/s102070100002.

[15] Sahbuddin Abdul Kadir, A. Sasongko, and M. Zulkifli, "Simple power analysis attack against elliptic curve cryptography processor on FPGA implementation," in *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, Jul. 2011, pp. 1–4. doi: 10.1109/ICEEI.2011.6021757.

[16] E. D. Mulder *et al.*, "Electromagnetic Analysis Attack on an FPGA Implementation of an Elliptic Curve Cryptosystem," in *EUROCON 2005 - The International Conference on "Computer as a Tool,"* Nov. 2005, vol. 2, pp. 1879–1882. doi: 10.1109/EURCON.2005.1630348.

[17] C. D. Walter, "Sliding Windows Succumbs to Big Mac Attack," in *Cryptographic Hardware and Embedded Systems — CHES 2001*, May 2001, pp. 286–299. doi: 10.1007/3-540-44709-1_24.

[18] J. Heyszl, S. Mangard, B. Heinz, F. Stumpf, and G. Sigl, "Localized Electromagnetic Analysis of Cryptographic Implementations," in *Topics in Cryptology – CT-RSA 2012*, vol. 7178, O. Dunkelman, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 231–244. doi: 10.1007/978-3-642-27954-6_15.

[19] A. Bauer, E. Jaulmes, E. Prouff, and J. Wild, "Horizontal and Vertical Side-Channel Attacks against Secure RSA Implementations," in *Topics in Cryptology – CT-RSA 2013*, Feb. 2013, pp. 1–17. doi: 10.1007/978-3-642-36095-4_1.

[20] A. Bauer, E. Jaulmes, E. Prouff, and J. Wild, "Horizontal Collision Correlation Attack on Elliptic Curves," in *Selected Areas in Cryptography -- SAC 2013*, Aug. 2013, pp. 553–570. doi: 10.1007/978-3-662-43414-7_28.

[21] I. Kabin, Z. Dyka, D. Kreiser, and P. Langendoerfer, "Horizontal address-bit DPA against montgomery *kP* implementation," in *2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dezember 2017, pp. 1–8. doi: 10.1109/RECONFIG.2017.8279800.

[22] I. Kabin, Z. Dyka, C. Wittke, D. Kreiser, P. Langendörfer, "Horizontal DEMA Attack as Low Cost Effective Mean to Reveal Keys", *Proc. 26th Crypto-Day,* (2017)

[23] Z. Dyka, I. Kabin, D. Kreiser, P. Langendörfer, „Horizontal DPA At-tacks: Low-Cost and High-Effective", *Proc. 26th Crypto-Day*, (2017)

[24] I. Kabin, Z. Dyka, D. Kreiser, P. Langendörfer, "Low-Cost and High Efficient Horizontal Attacks against ECDSA", *Proc. 27th Crypto-Day* 2018, (2018)

[25] E. Vogel, I. Kabin, Z. Dyka, P. Langendörfer, "Localized EMA as a Mean to Find Spatial and Time SCA Leakage Sources", *Proc. 27th Crypto-Day 2017*, (2018)

[26] I. Kabin, Z. Dyka, D. Kreiser, P. Langendörfer, "Attack against Montgomery kP Implementation: Horizontal Address-Bit DPA? ", *Proc. EUROMICRO Conference on Digital System Design* (DSD/SEAA 2017), (2017)

[27] I. Kabin, Z. Dyka, D. Kreiser, and P. Langendoerfer, "Horizontal Address-Bit DEMA against ECDSA," in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Feb. 2018, pp. 1–7. doi: 10.1109/NTMS.2018.8328695.

[28] I. Kabin, Z. Dyka, D. Klann, M. Aftowicz, and P. Langendoerfer, "Resistance of the Montgomery Ladder Against Simple SCA: Theory and Practice," *J Electron Test*, vol. 37, no. 3, pp. 289–303, Jun. 2021, doi: 10.1007/s10836-021-05951-3.

[29] I. Kabin, Z. Dyka, D. Klann, M. Aftowicz, and P. Langendoerfer, "FFT based Horizontal SCA Attack against ECC," in *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Apr. 2021, pp. 1–5. doi: 10.1109/NTMS49979.2021.9432665.

[30] J. Heyszl, A. Ibing, S. Mangard, F. De Santis, and G. Sigl, "Clustering Algorithms for Non-profiled Single-Execution Attacks on Exponentiations," in *Smart Card Research and Advanced Applications*, Cham, 2014, pp. 79–93. doi: 10.1007/978-3-319-08302-5_6.

[31] E. Nascimento and Ł. Chmielewski, "Applying Horizontal Clustering Side-Channel Attacks on Embedded ECC Implementations," in *Smart Card Research and Advanced Applications*, Cham, 2018, pp. 213–231. doi: 10.1007/978-3-319-75208-2_13.

[32] Y. Varabei, I. Kabin, Z. Dyka, D. Klann, and P. Langendoerfer, "Intelligent Clustering as a Means to Improve K-means Based Horizontal Attacks," in *2019 IEEE 30th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops)*, Sep. 2019, pp. 1–6. doi: 10.1109/PIMRCW.2019.8880831.

[33] I. Kabin, M. Aftowicz, D. Klann, Y. Varabei, Z. Dyka, D. Klann, P. Langendörfer, "Horizontal SCA Attack using Machine Learning Algorithms", Proc. *30th Crypto Day Matters* 2019, (2019)

[34] I. Kabin, M. Aftowicz, Y. Varabei, D. Klann, Z. Dyka, and P. Langendoerfer, "Horizontal Attacks using K-Means: Comparison with Traditional Analysis Methods," in *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Jun. 2019, pp. 1–7. doi: 10.1109/NTMS.2019.8763777.

[35] M. Aftowicz, I. Kabin, D. Klann, Y. Varabei, Z. Dyka, and P. Langendoerfer, "Horizontal SCA Attacks against *kP* Algorithm Using K-Means and PCA," in *2020 9th Mediterranean*

Conference on Embedded Computing (MECO), Jun. 2020, pp. 1–7. doi: 10.1109/MECO49872.2020.9134109.

[36] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Power Analysis Attacks of Modular Exponentiation in Smartcards," in *Cryptographic Hardware and Embedded Systems*, Aug. 1999, pp. 144–157. doi: 10.1007/3-540-48059-5_14.

[37] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Horizontal correlation analysis on exponentiation," in *In ICICS 2010, volume 6476 of LNCS*, 2010, pp. 46–61.

[38] B. Chevallier-Mames, M. Ciet, and M. Joye, "Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity," *IEEE Transactions on Computers*, vol. 53, no. 6, pp. 760–768, Jun. 2004, doi: 10.1109/TC.2004.13.

[39] P. Longa, "Accelerating the Scalar Multiplication on Elliptic Curve Cryptosystems over Prime Fields," *Cryptology ePrint Archive*, 2008, Accessed: Jul. 13, 2022. [Online]. Available: https://eprint.iacr.org/2008/100

[40] C. Giraud and V. Verneuil, "Atomicity Improvement for Elliptic Curve Scalar Multiplication," in *Proceedings of the 9th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Application*, Berlin, Heidelberg, 2010, pp. 80–101. doi: 10.1007/978-3-642-12510-2_7.

[41] I. Kabin, Z. Dyka, D. Kreiser, and P. Langendoerfer, "Unified field multiplier for ECC: Inherent resistance against horizontal SCA attacks," in *2018 13th International Conference on Design Technology of Integrated Systems In Nanoscale Era (DTIS)*, Apr. 2018, pp. 1–4. doi: 10.1109/DTIS.2018.8368560.

[42] D. Kreiser, Z. Dyka, I. Kabin, C. Wittke, P. Langendörfer, "HCCA against Montgomery kP Design", *Proc. 29th Crypto-Day 2018*, (2018)

[43] Z. Dyka, D. Kreiser, I. Kabin, and P. Langendoerfer, "Flexible FPGA ECDSA Design with a Field Multiplier Inherently Resistant against HCCA," in *2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dezember 2018, pp. 1–6. doi: 10.1109/RECONFIG.2018.8641730.

[44] K. Itoh, T. Izu, and M. Takenaka, "Address-Bit Differential Power Analysis of Cryptographic Schemes OK-ECDH and OK-ECDSA," in *Cryptographic Hardware and Embedded Systems - CHES 2002*, Aug. 2002, pp. 129–143. doi: 10.1007/3-540-36400-5_11.

[45] K. Itoh, T. Izu, and M. Takenaka, "A Practical Countermeasure against Address-Bit Differential Power Analysis," in *Cryptographic Hardware and Embedded Systems - CHES 2003*, Sep. 2003, pp. 382–396. doi: 10.1007/978-3-540-45238-6_30.

[46] M. Izumi, J. Ikegami, K. Sakiyama, and K. Ohta, "Improved countermeasure against Address-bit DPA for ECC scalar multiplication," in *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, Mar. 2010, pp. 981–984. doi: 10.1109/DATE.2010.5456907.

[47] C. Clavier and M. Joye, "Universal Exponentiation Algorithm A First Step towards Provable SPA-Resistance," in Cryptographic Hardware and Embedded Systems — CHES 2001, Berlin, Heidelberg, 2001, pp. 300–308. doi: 10.1007/3-540-44709-1_25.

[48] J.-S. Coron, "Resistance Against Differential Power Analysis For Elliptic Curve Cryptosystems," in *Cryptographic Hardware and Embedded Systems*, vol. 1717, Ç. K. Koç and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 292–302. doi: 10.1007/3-540-48059-5_25.

[49] R. Willi, A. Curiger, and P. Zbinden, "On Power-Analysis Resistant Hardware Implementations of ECC-Based Cryptosystems," in *2016 Euromicro Conference on Digital System Design (DSD)*, Aug. 2016, pp. 665–669. doi: 10.1109/DSD.2016.59.

[50] É. Brier and M. Joye, "Weierstraß Elliptic Curves and Side-Channel Attacks," in *Public Key Cryptography*, Feb. 2002, pp. 335–345. doi: 10.1007/3-540-45664-3_24.

[51] P. L. Montgomery, "Speeding the Pollard and elliptic curve methods of factorization," *Math. Comp.*, vol. 48, no. 177, pp. 243–264, 1987, doi: 10.1090/S0025-5718-1987-0866113-7.

[52] M. Joye and S.-M. Yen, "The Montgomery Powering Ladder," in *Cryptographic Hardware and Embedded Systems - CHES 2002*, Aug. 2002, pp. 291–302. doi: 10.1007/3-540-36400-5_22.

[53] J. Fan, X. Guo, E. De Mulder, P. Schaumont, B. Preneel, and I. Verbauwhede, "State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures," in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Jun. 2010, pp. 76–87. doi: 10.1109/HST.2010.5513110.

[54] M. Azouaoui, R. Poussier, and F.-X. Standaert, "Fast Side-Channel Security Evaluation of ECC Implementations," in *Constructive Side-Channel Analysis and Secure Design*, Cham, 2019, pp. 25–42. doi: 10.1007/978-3-030-16350-1_3.

[55] J. Samotyja and K. Lemke-Rust, "Practical Results of ECC Side Channel Countermeasures on an ARM Cortex M3 Processor," in *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security*, New York, NY, USA, Oct. 2016, pp. 27–35. doi: 10.1145/2996366.2996371.

[56] R. Bellman and E. G. Straus, "Addition chains of vectors (problem 5125)," *The American Mathematical Monthly*, vol. 71, no. 7, pp. 806–808, 1964, doi: 10.2307/2310929.

[57] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," in *Advances in Cryptology*, 1985, pp. 10–18.

[58] J.-S. Coron and L. Goubin, "On Boolean and Arithmetic Masking against Differential Power Analysis," in *Cryptographic Hardware and Embedded Systems — CHES 2000*, Berlin, Heidelberg, 2000, pp. 231–237. doi: 10.1007/3-540-44499-8_18.

[59] E. Oswald and M. Aigner, "Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks," in *Cryptographic Hardware and Embedded Systems — CHES 2001*, May 2001, pp. 39–50. doi: 10.1007/3-540-44709-1_5.

[60] F. Madlener, M. Sötttinger, and S. A. Huss, "Novel hardening techniques against differential power analysis for multiplication in GF(2^n)," in *2009 International Conference on Field-Programmable Technology*, Dec. 2009, pp. 328–334. doi: 10.1109/FPT.2009.5377676.

[61] N. Mentens, B. Gierlichs, and I. Verbauwhede, "Power and Fault Analysis Resistance in Hardware through Dynamic Reconfiguration," in *Cryptographic Hardware and Embedded Systems – CHES 2008*, Aug. 2008, pp. 346–362. doi: 10.1007/978-3-540-85053-3_22.

[62] Z. Dyka, P. Langendörfer, "Device and method for multiplication for impeding side-channel attacks", EP3215931/ US10474431, 2014; granted: 2018 Europe, 2019 US

[63] N. Pirotte, J. Vliegen, L. Batina, and N. Mentens, "Design of a Fully Balanced ASIC Coprocessor Implementing Complete Addition Formulas on Weierstrass Elliptic Curves," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, Aug. 2018, pp. 545–552. doi: 10.1109/DSD.2018.00095.

[64] L. Benini, A. Macii, E. Macii, E. Omerbegovic, M. Poncino, and F. Pro, "A novel architecture for power maskable arithmetic units," in *Proceedings of the 13th ACM Great Lakes symposium on VLSI*, New York, NY, USA, Apr. 2003, pp. 136–140. doi: 10.1145/764808.764845.

[65] L. Benini, E. Omerbegovic, A. Macii, M. Poncino, E. Macii, and F. Pro, "Energy-aware design techniques for differential power analysis protection," in *Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)*, Jun. 2003, pp. 36–41. doi: 10.1145/775832.775845.

[66] Z. Dyka, C. Wittke, and P. Langendoerfer, "Clockwise Randomization of the Observable Behaviour of Crypto ASICs to Counter Side Channel Attacks," in *2015 Euromicro Conference on Digital System Design*, Aug. 2015, pp. 551–554. doi: 10.1109/DSD.2015.40.

[67] X. Fan, S. Peter, and M. Krstic, "GALS design of ECC against side-channel attacks — A comparative study," in *2014 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Sep. 2014, pp. 1–6. doi: 10.1109/PATMOS.2014.6951905.

[68] J.-S. Coron and I. Kizhvatov, "An Efficient Method for Random Delay Generation in Embedded Software," in *Cryptographic Hardware and Embedded Systems - CHES 2009*, Springer, Berlin, Heidelberg, 2009, pp. 156–170. doi: 10.1007/978-3-642-04138-9_12.

[69] European Patent Application No. 22168962.3, Dummy partial multiplication of zero-operands as a low-cost means for reducing the success of SCA attacks against ECC, Unpublished (filing date Dec. 28, 2021)

[70] P. Kocher, J. Jaffe, and B. Jun, “Differential Power Analysis,” in *Advances in Cryptology — CRYPTO’ 99*, vol. 1666, Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397. doi: 10.1007/3-540-48405-1_25.

[71] N. Homma, A. Miyamoto, T. Aoki, A. Satoh, and A. Shamir, “Collision-Based Power Analysis of Modular Exponentiation Using Chosen-Message Pairs,” in *Cryptographic Hardware and Embedded Systems – CHES 2008*, Aug. 2008, pp. 15–29. doi: 10.1007/978-3-540-85053-3_2.

[72] P.-A. Fouque and F. Valette, “The Doubling Attack – Why Upwards Is Better than Downwards,” in *Cryptographic Hardware and Embedded Systems - CHES 2003*, Sep. 2003, pp. 269–280. doi: 10.1007/978-3-540-45238-6_22.

[73] H. Sakamoto, Y. Li, K. Ohta, and K. Sakiyama, “Fault Sensitivity Analysis Against Elliptic Curve Cryptosystems,” in *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, Sep. 2011, pp. 11–20. doi: 10.1109/FDTC.2011.17.

[74] E. Alpirez Bock, “SCA resistent implementation of the Montgomery kP-algorithm”, Accessed: Apr. 20, 2017. [Online]. Available: https://opus4.kobv.de/opus4-btu/frontdoor/index/index/docId/3628 (last accessed 01.11.2022).

[75] Z. Dyka, E. A. Bock, I. Kabin, and P. Langendoerfer, “Inherent Resistance of Efficient ECC Designs against SCA Attacks,” in *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Nov. 2016, pp. 1–5. doi: 10.1109/NTMS.2016.7792457.

[76] B. Ansari and M. A. Hasan, “High-Performance Architecture of Elliptic Curve Scalar Multiplication,” *IEEE Transactions on Computers*, vol. 57, no. 11, pp. 1443–1453, Nov. 2008, doi: 10.1109/TC.2008.133.

[77] Z. Dyka and P. Langendoerfer, “Area efficient hardware implementation of elliptic curve cryptography by iteratively applying Karatsuba’s method,” in *Design, Automation and Test in Europe*, Mar. 2005, pp. 70-75 Vol. 3. doi: 10.1109/DATE.2005.67.

[78] P. Langendörfer, Z. Dyka, P. Steffen, “Method and apparatus for calculating a polynomial multiplication, in particular for elliptic curve cryptography, ” US8477935B2, 2005, granted: 2013 US

[79] E. A. Bock and Z. Dyka, “Vulnerability assessment of an IHP ECC implementation”, Accessed: Jul. 13, 2020. [Online]. Available: https://opus4.kobv.de/opus4-btu/frontdoor/index/index/docId/3490 (last accessed 01.11.2022).

[80] TAMPRES Project, https://www.tampres.eu/project.html (last accessed 01.11.2022).

[81] I. Kabin, Z. Dyka, D. Klann, J. Schaeffner, and P. Langendoerfer, “On the Complexity of Attacking Commercial Authentication Products,” *Microprocessors and Microsystems*, vol. 80, p. 103480, Feb. 2021, doi: 10.1016/j.micpro.2020.103480.

[82] Synopsys PrimeTime Static Timing Analysis. https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html (last accessed 01.11.2022).

[83] A. Sosa, I. Kabin, Z. Dyka, D. Klann, P. Langendörfer, “On the Impact of the Sampling Rate on the Success of Horizontal DEMA Attack”, Proc. 31st Crypto Day Matters 2019, (2019)

[84] H. Ma, J. He, Y. Liu, Y. Zhao, and Y. Jin, “CAD4EM-P: Security-Driven Placement Tools for Electromagnetic Side Channel Protection,” in *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, Dec. 2019, pp. 1–6. doi: 10.1109/AsianHOST47458.2019.9006705.

[85] J. He, H. Ma, X. Guo, Y. Zhao, and Y. Jin, “Design for EM Side-Channel Security through Quantitative Assessment of RTL Implementations,” in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2020, pp. 62–67. doi: 10.1109/ASP-DAC47756.2020.9045426.

[86] H. Ma, J. He, Y. Liu, L. Liu, Y. Zhao, and Y. Jin, “Security-Driven Placement and Routing Tools for Electromagnetic Side-Channel Protection,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1077–1089, Jun. 2021, doi: 10.1109/TCAD.2020.3024938.

[87] Cadence: https://www.cadence.com/ (last accessed 17.11.2018).

[88] Tektronix: AC Current Probes Datasheet. https://www.tek.com/datasheet/current-probe/ct1-ct2-ct6 (last accessed 01.11.2022).

[89] Riscure: Current Probe. Quick Start Guide (Current Probe 1 - QSG 1.0 2015) https://riscureprodstorage.blob.core.windows.net/production/2017/07/currentprobe-qsg-1.0-released.pdf (last accessed 01.11.2022).

[90] HD Communications Corp.: HD24248. 0.1 – 2500MHz Low Noise Amplifier http://www.rfcomp.com/bbs/downPdf.php?filename=HD24248.pdf (last accessed 01.11.2022).

[91] Langer EMV-Technik. https://www.langer-emv.de/en/index (last accessed 01.11.2022).

[92] Riscure: EM Capacitor Probe Set. https://getquote.riscure.com/en/quote/2101063/em-capacitor-probe-set.htm (last accessed 01.11.2022).

[93] Langer MFA-R 0.2-75 Near-Field Micro Probe 1 MHz up to 1 GHz. https://www.langer-emv.de/en/product/mfa-active-1mhz-up-to-6-ghz/32/mfa-r-0-2-75-near-field-micro-probe-1-mhz-up-to-1-ghz/854 (last accessed 01.11.2022).

[94] Langer LF-B 3 H-Field Probe 100 kHz up to 50 MHz https://www.langer-emv.de/en/product/lf-passive-100-khz-up-to-50-mhz/36/lf-b-3-h-field-probe-100-khz-up-to-50-mhz/3 (last accessed 01.11.2022).

[95] Langer ICR HH150-27 set Near-Field Microprobe 1.5 MHz - 6 GHz https://www.langer-emv.de/en/product/near-field-microprobe-sets-icr-hh-h-field/26/icr-hh150-27-set-near-field-microprobe-1-5-mhz-6-ghz/757 (last accessed 01.11.2022).

[96] Langer ICR HV150-27 set Near-Field Microprobe 1.5 MHz - 6 GHz https://www.langer-emv.de/en/product/near-field-microprobe-sets-icr-hv-h-field/40/icr-hv150-27-set-near-field-microprobe-1-5-mhz-6-ghz/761 (last accessed 01.11.2022).

[97] C. Wittke, I. Kabin, D. Klann, Z. Dyka, A. Datsuk, and P. Langendoerfer, "Horizontal DEMA Attack as the Criterion to Select the Best Suitable EM Probe," 1181, 2018. Accessed: Nov. 01, 2022. [Online]. Available: https://eprint.iacr.org/2018/1181

[98] C. Wittke, I. Kabin, D. Kreiser, Z. Dyka, P. Langendörfer, "Selecting the Best Suitable EM Probe Using Horizontal DEMA Attack", *Proc. 26th Crypto-Day*, (2017)

[99] I. Kabin, Z. Dyka, D. Kreiser, P. Langendörfer, "Improving DEMA Attack Results using Different Compression Methods", *Proc. 27th Crypto-Day 2018*, (2018)

[100] B. L. Welch, "The generalisation of student's problems when several different population variances are involved," *Biometrika*, vol. 34, no. 1–2, pp. 28–35, 1947, doi: 10.1093/biomet/34.1-2.28.

[101] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[102] R.A Fisher, "Statistical methods for research workers", (5th ed.). : Oliver and Boyd: Edinburgh, 1934.

[103] M. Aftowicz, I. Kabin, Z. Dyka, P. Langendörfer, „Clustering versus Sta-tistical Analysis for SCA: when Machine Learning is Better",10th Mediterranean Conference on Embedded Computing (MECO), 2021, pp. 1-5, doi: 10.1109/MECO52532.2021.9460161

[104] S. Mangard, "A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion," in *Information Security and Cryptology — ICISC 2002*, Berlin, Heidelberg, 2003, pp. 343–358. doi: 10.1007/3-540-36552-4_24.

[105] I. Kabin, Z. Dyka, M. Aftowicz, D. Klann, and P. Langendoerfer, "Resistance of the Montgomery *kP* Algorithm against Simple SCA: Theory and Practice," in *2020 IEEE Latin-American Test Symposium (LATS)*, Mar. 2020, pp. 1–6. doi: 10.1109/LATS49555.2020.9093678.

[106] I. Kabin, Z. Dyka, and P. Langendoerfer, "Atomicity and Regularity Principles Do Not Ensure Full Resistance of ECC Designs against Single-Trace Attacks," *Sensors*, vol. 22, no. 8, Art. no. 8, Jan. 2022, doi: 10.3390/s22083083.

[107] I. Kabin, E. Vogel, Z. Dyka, P. Langendoerfer, “EMA as a mean to find spatial and time SCA leakage sources”, Technical Demonstrations Session of the 2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig17), 10.1109/RECONFIG.2017.8279818

[108] I. Kabin, D. Klann, Z. Dyka, A. Datsuk, P. Langendoerfer, “DEMO: Demonstrating Horizontal Attacks using IHP’s Side Channel Analysis Tool”, Technical Demonstrations Session of the 2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig18) https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8641721

[109] I. Kabin, Z. Dyka, and P. Langendoerfer, “Automated Simple Analysis Attack,” in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, Jun. 2020, pp. 1–4. doi: 10.1109/MECO49872.2020.9134160.

[110] I. Kabin, E. Alpirez Bock, C. Wittke, D. Kreiser, Z. Dyka, P. Langen-dörfer, „On the Influence of Hardware Technologies on the Vulnera-bility of Protected ECC Implementations“,Proc. Session on Work in Progress, *19th EUROMICRO Conference on Digital Systems Design* (DSD 2016), (2016)

[111] I. Kabin, Z. Dyka, D. Kreiser, and P. Langendoerfer, “Evaluation of resistance of ECC designs protected by different randomization countermeasures against horizontal DPA attacks,” in *2017 IEEE East-West Design Test Symposium (EWDTS)*, Sep. 2017, pp. 1–7. doi: 10.1109/EWDTS.2017.8110037.

[112] IHP - Innovations for High Performance Microelectronics: www.ihp-microelectronics.com (last accessed 01.11.2022).

[113] Z. Dyka, I. Kabin, D. Klann, F. Vater, P. Langendörfer, “Caution: GALS-ification as a Means against SCA Attacks”, *Proc. 17th IEEE East-West Design & Test Symposium* (EWDTS 2019), 97 (2019)

[114] Certicom Corp., “Standards for Efficient Cryptography 2 (SEC 2)” https://www.secg.org/sec2-v2.pdf (last accessed 01.11.2022).

[115] N. Pirotte, “Design of a fully balanced ASIC coprocessor implementing complete addition formulas on Weierstrass elliptic curves”, Master Thesis, 2018. https://documentserver.uhasselt.be/handle/1942/27052 (last accessed 01.11.2022).

[116] L. Batina, J. Hogenboom, N. Mentens, J. Moelans, and J. Vliegen, “Side-channel evaluation of FPGA implementations of binary Edwards curves,” in *2010 17th IEEE International Conference on Electronics, Circuits and Systems*, Dec. 2010, pp. 1248–1251. doi: 10.1109/ICECS.2010.5724745.

[117] N. Pirotte, *VHDL source code for the design implemented in Master Thesis*. 2018. Accessed: Dec. 20, 2020. [Online]. Available: https://github.com/NielsPirotte/MasterThesis_Niels_Pirotte (last accessed 01.11.2022).

[118] I. Kabin, Z. Dyka, D. Klann, N. Mentens, L. Batina, and P. Langendoerfer, “Breaking a fully Balanced ASIC Coprocessor Implementing Complete Addition Formulas on Weierstrass Elliptic Curves,” in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, Aug. 2020, pp. 270–276. doi: 10.1109/DSD51259.2020.00051.

[119] IHP – SiGe: C BiCMOStechnologies. https://www.ihp-microelectronics.com/services/research-and-prototyping-service/mpw-prototyping-service/sigec-bicmos-technologies (last accessed 01.11.2022).

[120] I. Kabin, Z. Dyka, P. Langendörfer, „Influence of Multiplier on the Security of Elliptic Curve Cryptography Design“, Proc. 25rd Crypto-Day, (2016)

[121] I. Kabin, Z. Dyka, D. Klann, and P. Langendoerfer, “Horizontal DPA Attacks against ECC: Impact of Implemented Field Multiplication Formula,” in *2019 14th International Conference on Design Technology of Integrated Systems In Nanoscale Era (DTIS)*, Apr. 2019, pp. 1–6. doi: 10.1109/DTIS.2019.8735011.

[122] “A. Karatsuba and Y. Ofman, ‘Multiplication of Many-Digital Numbers by Automatic Computers,’ Doklady Akademii Nauk SSSR, Vol. 145, No. 2, 1962, pp. 293-294. - References https://www.researchgate.net/publication/234346907_Multiplication_of_Multidigit_Numbers_on_Automata (last accessed 01.11.2022).

[123] Dyka, Z. Analysis and prediction of area- and energy-consumption of optimized polynomial multipliers in hardware for arbitrary GF(2^n) for elliptic curve cryptography. Dissertation thesis, BTU Cottbus-Senftenberg (2013). https://opus4.kobv.de/opus4-btu/frontdoor/index/index/docId/2634 (last accessed 01.11.2022).

[124] Z. Dyka, P. Langendoerfer, and F. Vater, "Combining Multiplication Methods with Optimized Processing Sequence for Polynomial Multiplier in GF(2^k)," in *Research in Cryptology*, Berlin, Heidelberg, 2012, pp. 137–150. doi: 10.1007/978-3-642-34159-5_10.

[125] M. Stöttinger, F. Madlener, and S. A. Huss, "Procedures for Securing ECC Implementations Against Differential Power Analysis Using Reconfigurable Architectures," in *Dynamically Reconfigurable Systems*, M. Platzner, J. Teich, and N. Wehn, Eds. Springer Netherlands, 2010, pp. 395–415. doi: 10.1007/978-90-481-3485-4_19.

[126] S. Winograd, *Arithmetic Complexity of Computations*. Society for Industrial and Applied Mathematics, 1980. doi: 10.1137/1.9781611970364.

[127] I. Kabin, Z. Dyka, D. Kreiser, and P. Langendoerfer, "Methods for Increasing the Resistance of Cryptographic Designs Against Horizontal DPA Attacks," in *Information and Communications Security*, Dec. 2017, pp. 225–235. doi: 10.1007/978-3-319-89500-0_20.

[128] Z. Dyka, E. Alpirez Bock, I. Kabin, P. Langendörfer, „Revisiting Random Permutation of Calculation Steps of a Field Multiplication as a DPA Countermeasure", *Proc. 25th Crypto-Day*, (2016)

[129] I. Kabin, D. Kreiser, Z. Dyka, P. Langendörfer , "FPGA Implementa-tion of ECC: Low-Cost Countermeasure against Horizontal Bus and Address-Bit SCA", Proc. International Conference on Reconfigurable Computing and FPGAs (ReConFig 2018), (2018)

[130] I. Kabin, Z. Dyka, D. Kreiser, P. Langendörfer, "Low-Cost Countermeasure Against Horizontal Bus and Address-Bit SCA", Proc. 29th Crypto-Day 2018, (2018)

[131] I. Kabin, Z. Dyka, D. Klann, and P. Langendoerfer, "Methods increasing inherent resistance of ECC designs against horizontal attacks," *Integration*, vol. 73, pp. 50–67, Jul. 2020, doi: 10.1016/j.vlsi.2020.03.001.

[132] I. Kabin, Z. Dyka, D. Klann, P. Langendörfer, "Influence of Synopsys Design Compiler Options on the Success of Horizontal DPA Attacks", *Proc. 22nd EUROMICRO Conference on Digital System Design and Software Engineering and Advanced Applications* - Session on Work in Progress (Euromicro DSD & SEAA 2019), (2019)

[133] I. Kabin, Z. Dyka, D. Klann, and P. Langendoerfer, "Horizontal Attacks Against ECC: From Simulations to ASIC," in *Computer Security*, Cham, 2020, pp. 64–76. doi: 10.1007/978-3-030-42051-2_5.

[134] D. Klann, M. Aftowicz, I. Kabin, Z. Dyka, and P. Langendoerfer, "Integration and Implementation of four different Elliptic Curves in a single high-speed Design considering SCA," in *2020 15th Design & Technology of Integrated Systems in Nanoscale Era* (DTIS), Apr. 2020, pp. 1–2. doi: 10.1109/DTIS48698.2020.9081300.

[135] I. Kabin, D. Kreiser, Z. Dyka, P. Langendörfer, "A Secure Scalable Dual-Field Multiplier for ECC", Proc. *EUROMICRO Conference on Digital System Design and Software Engineering and Advanced Applications* (DSD & SEAA 2018) - Session on Work in Progress, (2018)

[136] I. Kabin, Z. Dyka, D. Klann, and P. Langendoerfer, "Fast and Secure Unified Field Multiplier for ECC Based on the 4-Segment Karatsuba Multiplication," in *2019 IEEE East-West Design Test Symposium (EWDTS)*, Sep. 2019, pp. 1–6. doi: 10.1109/EWDTS.2019.8884393.

[137] I. Kabin, D. Klann, Z. Dyka, and P. Langendoerfer, "Fast Dual-Field ECDSA Accelerator with Increased Resistance against Horizontal SCA Attacks," in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, Jul. 2021, pp. 273–280. doi: 10.1109/CSR51186.2021.9527912.

[138] Z. Dyka, I. Kabin, D. Klann, and P. Langendoerfer, "Multiplier as a Mean for Reducing Vulnerability of Atomic Patterns to Horizontal Address-Bit Attacks," in *2021 10th Mediterranean Conference on Embedded Computing (MECO)*, Jun. 2021, pp. 1–6. doi: 10.1109/MECO52532.2021.9460158.

[139] I. Kabin, Z. Dyka, D. Klann, and P. Langendoerfer, “EC Scalar Multiplication: Successful Simple Address-Bit SCA Attack against Atomic Patterns,” in *2021 IEEE 22nd Latin American Test Symposium (LATS)*, Oct. 2021, pp. 1–2. doi: 10.1109/LATS53581.2021.9651877.

[140] I. Kabin, A. Sosa, Z. Dyka, D. Klann, and P. Langendoerfer, “On the Influence of the FPGA Compiler Optimization Options on the Success of the Horizontal Attack,” in 2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig), Dec. 2019, pp. 1–5. doi: 10.1109/ReConFig48160.2019.8994807.

[141] Z. Dyka et al., “On the SCA Resistance of Crypto IP Cores,” in 2022 IEEE 23rd Latin American Test Symposium (LATS), Sep. 2022, pp. 1–2. doi: 10.1109/LATS57337.2022.9937007.

[142] Dyka, Z., Vater, F., Wittke, C., Datsuk, A. and Langendoerfer, P. 2014. Using Supply Voltage Metal Layers as Low Cost Means to Hinder Several Types of Physical Attacks. In *Proceedings of the International Conference in Information Security and Digital Forensics (ISDF2014)*, 54–62.

[143] D. Goodin, “Post-quantum encryption contender is taken out by single-core PC and 1 hour,” *Ars Technica*, Aug. 02, 2022. https://arstechnica.com/information-technology/2022/08/sike-once-a-post-quantum-encryption-contender-is-koed-in-nist-smackdown/ (last accessed 01.11.2022)

Appendix 1

Table 1 shows the operation sequence implemented in the main loop of the IHP ***kP*** design for ECs over $GF(2^n)$. In the main loop a single key bit value k_i of the scalar k is processed.

Table 1. Operation flow in the main loop iterations in the IHP *kP* design for ECs over $GF(2^n)$.

Operation	$k_i = 0$	$k_i = 1$
M1	$Z_2 \leftarrow X_1 \cdot Z_2$ (or $X_4 \leftarrow X_2 \cdot Z_1$)	$Z_1 \leftarrow X_1 \cdot Z_2$ (or $X_4 \leftarrow X_2 \cdot Z_1$)
*Sq*1	$X_3 \leftarrow Z_1^2$	$X_3 \leftarrow Z_2^2$
*Sq*2	$a = Z_1^4$	$a = Z_2^4$
M2	$X_1 \leftarrow X_2 \cdot Z_1$ (or $Z_2 \leftarrow X_1 \cdot Z_2$)	$X_2 \leftarrow X_2 \cdot Z_1$ (or $Z_1 \leftarrow X_1 \cdot Z_2$)
	$Z_1 \leftarrow a$	$Z_2 \leftarrow a$
*Sq*3	$X_2 \leftarrow X_1^2$	$X_1 \leftarrow X_2^2$
M3	$X_1 \leftarrow b \cdot Z_1$	$X_2 \leftarrow b \cdot Z_2$
*Add*1	$a = Z_2 + X_1$ (or $a = X_4 + Z_2$)	$a = Z_1 + X_2$ (or $a = X_4 + Z_1$)
M4	$Z_1 \leftarrow X_1 \cdot Z_2$ (or $Z_1 \leftarrow X_4 \cdot Z_2$)	$Z_2 \leftarrow X_2 \cdot Z_1$ (or $Z_2 \leftarrow X_4 \cdot Z_1$)
*Sq*4	$Z_2 \leftarrow a^2$	$Z_1 \leftarrow a^2$
M5	$m = x \cdot Z_2$	$m = x \cdot Z_1$
*Sq*5,*Add*2	$X_1 \leftarrow X_2^2 + X_1$	$X_2 \leftarrow X_1^2 + X_2$
M6	$Z_1 \leftarrow X_3 \cdot X_2$	$Z_2 \leftarrow X_3 \cdot X_1$
*Add*3	$X_2 \leftarrow m + Z_1$	$X_1 \leftarrow m + Z_2$

The main loop consists of a sequence of 6 field multiplications (denoted **M**1, ..., **M**6), 5 field squaring operations (denoted *Sq1*, ..., *Sq5*), 3 field additions (denoted *Add1*, *Add2*, *Add3*) and many *write-to-register* operations (denoted with "←").

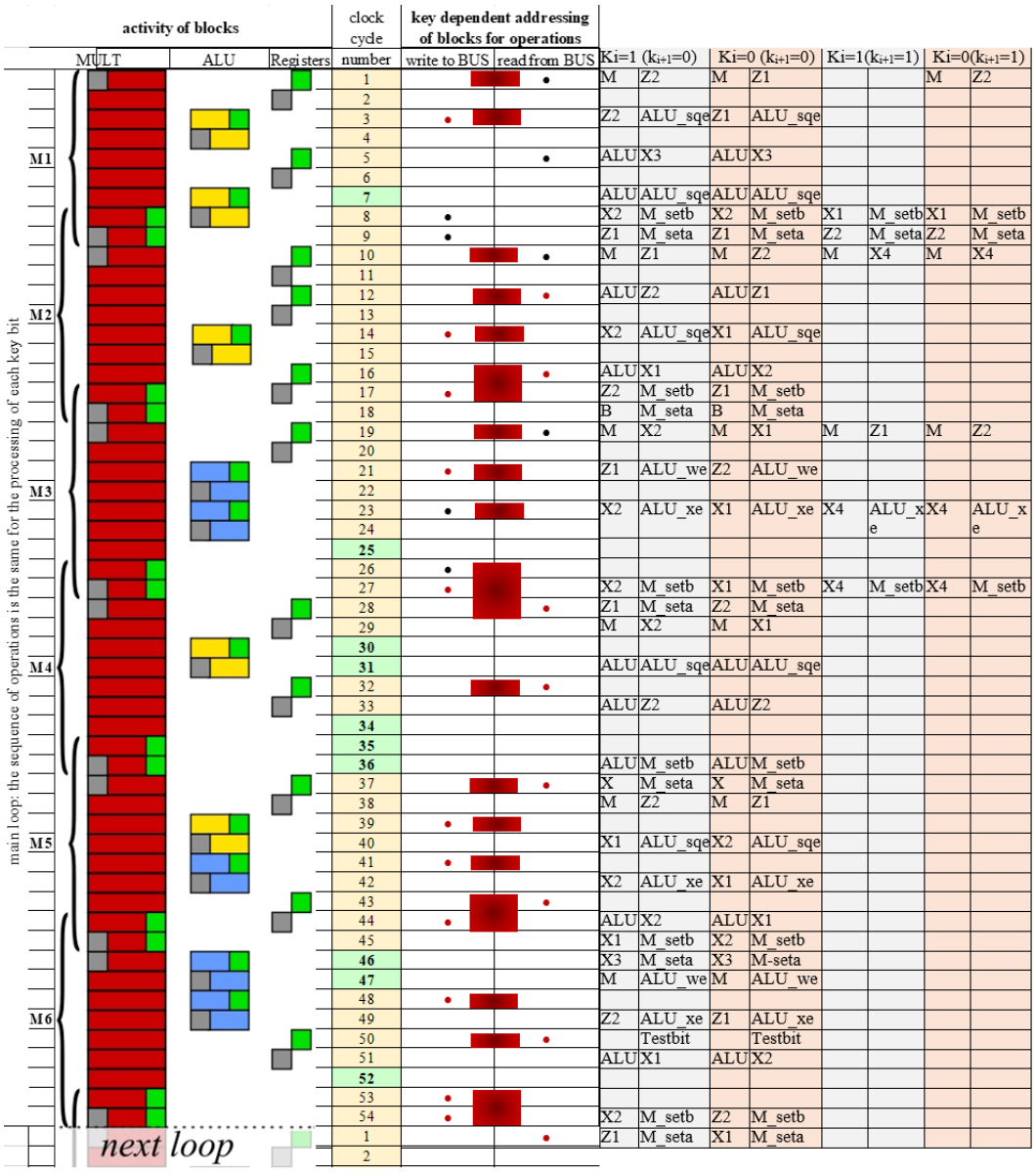
The field addition is a bitwise XOR operation. The field squaring of an element $A(t) = a_{n-1}t^{n-1} + a_{n-2}t^{n-2} + \ldots \quad t_0 \in GF(2^n)$ with an irreducible polynomial $f(t)$ is performed corresponding to the following formula using the binary representation of $A(t) = a_{n-1} \ldots \quad \sum_{i=0}^{n-1} a_0 \cdot 2^i : (A(t))^2 = \left(\sum_{i=0}^{n-1} a_i \cdot 2^{2i} \right) \bmod f(t) = a_{n-1} 0 a_{n-2} 0 \ldots \quad \bmod f(t) \cdot$

It is an easy operation requiring only a single clock cycle. It is performed in parallel to other multiplications accelerating the ***kP*** executions significantly. The field addition and field squaring operations are performed in the block called ALU.

Only 7 registers are used in the IHP design: X_1, X_2, Z_1, Z_2, X_3, X_4 and x. The register x contains the affine x-coordinate of the input point ***P***.

Please note that a and m in Table 1 are not physical registers. They are variables that we use here to denote some intermediate values. In our implementation all squaring operations, additions and write-to-register operations are realized in parallel to field multiplications.

The following diagram shows the processing sequence of the main loop in the IHP basic ***kP*** design, i.e. implementation details including the key-dependent addressing of the blocks for write-to-bus and read-from-bus operations. Due to the structure of the design and the high level of pipelining achieved, a key bit processing requires only 54 clock cycles and is performed always using the same operation sequence. The clock cycles marked with red rectangles in the columns "key-dependent addressing of blocks" are strong SCA leakage sources corresponding to our analysis results.



	Ki=1 (k_{i+1}=0)		Ki=0 (k_{i+1}=0)		Ki=1(k_{i+1}=1)		Ki=0(k_{i+1}=1)	
1	M	Z2	M	Z1			M	Z2
2								
3	Z2	ALU_sqe	Z1	ALU_sqe				
4								
5	ALU	X3	ALU	X3				
6								
7	ALU	ALU_sqe	ALU	ALU_sqe				
8	X2	M_setb	X2	M_setb	X1	M_setb	X1	M_setb
9	Z1	M_seta	Z1	M_seta	Z2	M_seta	Z2	M_seta
10	M	Z1	M	Z2	M	X4	M	X4
11								
12	ALU	Z2	ALU	Z1				
13								
14	X2	ALU_sqe	X1	ALU_sqe				
15								
16	ALU	X1	ALU	X2				
17	Z2	M_setb	Z1	M_setb				
18	B	M_seta	B	M_seta				
19	M	X2	M	X1	M	Z1	M	Z2
20								
21	Z1	ALU_we	Z2	ALU_we				
22								
23	X2	ALU_xe	X1	ALU_xe	X4	ALU_xe	X4	ALU_xe
24								
25								
26								
27	X2	M_setb	X1	M_setb	X4	M_setb	X4	M_setb
28	Z1	M_seta	Z2	M_seta				
29	M	X2	M	X1				
30								
31	ALU	ALU_sqe	ALU	ALU_sqe				
32								
33	ALU	Z2	ALU	Z2				
34								
35								
36	ALU	M_setb	ALU	M_setb				
37	X	M_seta	X	M_seta				
38	M	Z2	M	Z1				
39								
40	X1	ALU_sqe	X2	ALU_sqe				
41								
42	X2	ALU_xe	X1	ALU_xe				
43								
44	ALU	X2	ALU	X1				
45	X1	M_setb	X2	M_setb				
46	X3	M_seta	X3	M-seta				
47	M	ALU_we	M	ALU_we				
48								
49	Z2	ALU_xe	Z1	ALU_xe				
50		Testbit		Testbit				
51	ALU	X1	ALU	X2				
52								
53								
54	X2	M_setb	Z2	M_setb				
1	Z1	M_seta	X1	M_seta				
2								

Appendix 2

Algorithm A1: Montgomery ladder for point multiplication with random execution order, as it proposed in [116]

Input: P, $m=\{m_{t-1}, \ldots, m_0\}$, $m_{t-1}=1$, random bits $r_{t-2}, \ldots, r_0$
Output: $R=mP$

1. $R_0=P$
2. $R_1=2P$
3. **for** $i=t-2$ *to* 0 **do**
4. **if** $m_i=1$ **then**
5. **if** $r_i=0$ **then**
6. $T_0=R_0+R_1;\ T_1=2R_1;$
7. $R_0=T_0;\ R_1=T_1;$
8. **else**
9. $T_1=2R_1;\ T_0=R_0+R_1;$
10. $R_0=T_0;\ R_1=T_1;$
11. **end**
12. **else**
13. **if** $r_i=0$ **then**
14. $T_1=R_0+R_1;\ T_0=2R_0;$
15. $R_0=T_0;\ R_1=T_1;$
16. **else**
17. $T_0=2R_0;\ T_1=R_0+R_1;$
18. $R_0=T_0;\ R_1=T_1;$
19. **end**
20. **end**
21. {R_1- R_0 *remains invariant*}
22. **end**
23. $R=R_0$