Technische Universität Berlin

Fakultät für Elektrotechnik und Informatik
Institut für Softwaretechnik und Theoretische Informatik
Lehrstuhl für Security in Telecommunications

Why Cryptography Should Not Rely on Physical Attack Complexity

vorgelegt von
Juliane I. Krämer (Dipl. Math. oec.)
aus Detmold

von der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
genehmigte Dissertation

Promotionsausschuss:

Vorsitzende:	Prof. Anja Feldmann, Ph.D., Technische Universität Berlin
Gutachter:	Prof. Dr. Jean-Pierre Seifert, Technische Universität Berlin
Gutachter:	Prof. Dr. Johannes Buchmann, Technische Universität Darmstadt
Gutachterin:	Prof. Dr. Tanja Lange, Technische Universiteit Eindhoven

Tag der wissenschaftlichen Aussprache: 23. März 2015

Berlin 2015
D83

Ich versichere an Eides statt, dass ich diese Dissertation selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Datum

Für meine Eltern

Zusammenfassung

Seitdem Mitte der Neunziger Jahre die ersten Seitenkanal- und Fehlerangriffe auf kryptographische Technologien vorgestellt wurden, werden kontinuierlich neue Möglichkeiten physikalischer Angriffe erforscht. Der Gefahr, die von diesen Angriffen ausgeht, wird begegnet, indem auf bekannte Angriffe reagiert wird und Gegenmaßnahmen zum Schutz vor ihnen implementiert werden. Bei physikalischen Angriffen, die zwar prinzipiell bekannt sind, jedoch noch nicht praktisch umgesetzt wurden, verhält es sich hingegen anders. Angriffe, deren Realisierung eine hohe physikalische Komplexität zugeschrieben wird, werden weniger ernst genommen. Das Vertrauen darauf, dass diese Angriffe aufgrund ihrer physikalischen Komplexität nicht möglich sein werden, führt dazu, dass auf keiner Ebene Gegenmaßnahmen für sie entwickelt werden. Dieses Vorgehen ist problematisch, wenn sich im Nachhinein durch die Realisierung solcher Angriffe die Einschätzung der Komplexität als falsch erweist.

In dieser Arbeit werden zwei praktische physikalische Angriffe präsentiert, deren Theorie bereits seit mehreren Jahren bekannt ist. Da diese Angriffe jedoch zuvor nicht erfolgreich praktisch umgesetzt wurden, wurde in ihnen keine Gefahr gesehen. Die Komplexität ihrer Durchführung wurde überschätzt. Zunächst stellen wir den photonischen Seitenkanal vor, der neben der zeitlichen die größtmögliche räumliche Auflösung bietet, aufgrund der hohen Kosten bei seiner ersten Anwendung jedoch bisher nicht ernst genommen wurde. Wir zeigen sowohl einfache als auch differentielle photonische Seitenkanalanalysen. Anschließend präsentieren wir einen Fehlerangriff auf paarungsbasierte Kryptographie, der aufgrund der Notwendigkeit zweier unabhängiger präziser Fehler in einer einzigen Paarungsberechnung bei der Entwicklung von Gegenmaßnahmen nicht berücksichtigt wurde. Wir zeigen, wie Angreifer mit Hilfe dieser physikalischen Angriffe geheimes Schlüsselmaterial symmetrischer und asymmetrischer Algorithmen ermitteln können. Anschließend präsentieren wir Gegenmaßnahmen auf Software- und Hardware-Ebene, mit deren Hilfe diesen neuen Angriffen zukünftig standgehalten werden kann.

Anhand der beiden vorgestellten Angriffe zeigt diese Arbeit, dass die Einschätzung physikalischer Angriffskomplexität fehlerhaft sein kann. Es ist daher falsch, auf sie zu vertrauen. Kryptographische Technologien sollten gegenüber sämtlichen physikalischen Angriffen geschützt werden, seien diese bereits praktisch umgesetzt oder nur theoretisch bekannt. Die Entwicklung von Gegenmaßnahmen erfordert nicht die erfolgreiche Durchführung praktischer Angriffe und sollte daher bereits erfolgen, sobald das Prinzip eines Seitenkanals oder eines Fehlerangriffs verstanden ist.

Abstract

Ever since the first side channel attacks and fault attacks on cryptographic devices were introduced in the mid-nineties, new possibilities of physical attacks have been consistently explored. The risk that these attacks pose is reduced by reacting to known attacks and by developing and implementing countermeasures against them. For physical attacks whose theory is known but which have not been conducted yet, however, the situation is different. Attacks whose physical realization is assumed to be very complex are taken less seriously. The trust that these attacks will not be realized due to their physical complexity means that no countermeasures are developed at all. This leads to unprotected devices once the assessment of the complexity turns out to be wrong.

This thesis presents two practical physical attacks whose theory is known for several years. Since neither attack has previously been successfully implemented in practice, however, they were not considered a serious threat. Their physical attack complexity has been overestimated and the implied security threat has been underestimated. First, we introduce the photonic side channel, which offers not only temporal resolution, but also the highest possible spatial resolution. Due to the high cost of its first realization, it has not been taken seriously. We show both simple and differential photonic side channel analyses. Then, we present a fault attack against pairing-based cryptography. Due to the need for at least two independent precise faults in a single pairing computation, it has also not been taken seriously. We show how attackers can reveal the secret key of symmetric as well as asymmetric cryptographic algorithms based on these physical attacks. We present countermeasures on the software and the hardware level, which help to prevent these attacks in the future.

Based on these two presented attacks, this thesis shows that the assessment of physical attack complexity is error-prone. Hence, cryptography should not rely on it. Cryptographic technologies have to be protected against all physical attacks, have they already been realized or not. The development of countermeasures does not require the successful execution of an attack but can already be carried out as soon as the principle of a side channel or a fault attack is understood.

Publications Related to this Thesis

The primary results of this work have been presented in the following publications:

- Blömer, Gomes da Silva, Günther, **Krämer**, Seifert: *A Practical Second-Order Fault Attack against a Real-World Pairing Implementation*. In Proceedings of Fault Tolerance and Diagnosis in Cryptography (FDTC), 2014, Busan, Korea
- **Krämer**, Kasper, Seifert: *The Role of Photons in Cryptanalysis*. In Proceedings of 19th Asia and South Pacific Design Automation Conference (ASP-DAC), 2014, Singapore
- **Krämer**, Nedospasov, Schlösser, Seifert: *Differential Photonic Emission Analysis*. In Proceedings of Constructive Side-Channel Analysis and Secure Design - Fourth International Workshop (COSADE), 2013, Paris, France
- Schlösser, Nedospasov, **Krämer**, Orlic, Seifert: *Simple Photonic Emission Analysis of AES*. Journal of Cryptographic Engineering, Springer-Verlag
- Schlösser, Nedospasov, **Krämer**, Orlic, Seifert: *Simple Photonic Emission Analysis of AES*. In Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES), 2012, Leuven, Belgium

Additionally, Juliane Krämer has authored the following publications:

- **Krämer**, Stüber, Kiss: *On the Optimality of Differential Fault Analyses on CLEFIA*. Cryptology ePrint Archive, Report 2014/572
- **Krämer**: *Anwendungen von identitätsbasierter Kryptographie*. SmartCard Workshop 2014, Darmstadt, Germany
- Michéle, **Krämer**, Seifert: *Structure-Based RSA Fault Attacks*. In Proceedings of 8th International Conference on Information Security Practice and Experience (ISPEC), 2012, Hangzhou, China
- **Krämer**, Nedospasov, Seifert: *Weaknesses in Current RSA Signature Schemes*. In Proceedings of 14th International Conference on Information Security and Cryptology (ICISC), 2011, Seoul, Korea

Contents

Chapter 1

Introduction

Cryptanalysis is the art and science of revealing secret information of cryptographic systems. Until approximately 20 years ago, cryptanalysts primarily utilized mathematics to analyze cryptographic schemes. In parallel to the development of more sophisticated ciphers, cryptanalysts were also forced to develop stronger analysis methods. To break the Caesar cipher, which was used 2000 years ago, a simple statistical analysis of a ciphertext was sufficient. The Vigenère cipher was also broken with statistical analysis, albeit in the 19th century. Today's cryptanalytic attacks are not as simple as a statistical analysis, but involve complex mathematical techniques: differential cryptanalysis, which is mainly applied to block ciphers, analyzes the relation between differences in plaintexts and differences in ciphertexts [26]. Linear cryptanalysis approximates algorithms and their non-linear operations, respectively, with linear functions to reveal information about the secret key [114]. Related-key attacks study the influence of key-scheduling algorithms on the strength of block ciphers [25]. These attacks are independent of the number of rounds the block cipher undergoes. Even AES-192 and AES-256 can be weakened with related-key attacks [28].

Despite the existence of these attacks, our current knowledge of mathematics and cryptography allows us to construct secure schemes that can withstand such attacks. The mathematical principles of today's ciphers are strong enough to securely protect sensitive data, and practical applications of modern ciphers are not threatened by these attacks.

The mathematical strength of an algorithm, however, is only one aspect of the security of a cryptosystem. The resistance against physical attacks is also an important consideration. These do not target the underlying mathematical principles, but the implementation of the cipher. Consequently, they are also called implementation attacks [96]. In addition to the mathematical cryptanalyses, these implementation attacks nowadays also form part of cryptanalysis and can, in turn, be divided into two groups (see Figure 1.1, which is based on [131]). The first group measures physical characteristics of the device performing the attacked cryptographic operations, without modifying the computation. These attacks are called passive attacks or side channel attacks. The first published side channel attack analyzed timing variations of several cryptographic algorithms and was published in 1996 [102]. It was shown that a secret exponent from a modular exponentiation, as used in RSA, can be revealed by successively analyzing the timing variations emerging from the value of the exponent bits. For a standard square-and-multiply algorithm, a 1-bit needs

more processing steps and is therefore more time-consuming than a 0-bit. It was also presented how timing variations can reveal information about modular reduction, which in turn reveals information about the size of the processed values. Interestingly, the underlying principle of side channel attacks was known long before 1996, at least to secret services: in 1952, a covert listening device was found in the Moscow embassy of the United States. It was a replica of the Great Seal of the United States, presented from Soviet youths to the U.S. ambassador in Moscow already in 1946. This device is known as the Thing, or the Great Seal bug. A radio beam was driven at the antenna from a transmitter outside the embassy. The secret information, i.e, the conversations inside the room, was revealed by analyzing the modulation in the reflected signal emanating from the bug [182]. Thus, with the Great Seal bug, secret information was extracted from physical signals 50 years before Kocher's pioneering publication. The second group of implementation attacks actively modifies the computation and alters operations by, e.g., randomly changing values [119], changing the sign of a value [35], or skipping instructions [13]. These effects can be achieved by various mechanisms [17]. Such attacks are called invasive attacks, active attacks, or fault attacks[1]. In 1997, the first fault attack was published [37]. The attack targets the RSA signature scheme. It was shown that the RSA modulus can be factorized if an attacker induces a fault into the computation of one of the two parts of the signature generation when the RSA CRT version is used. With a single faulty signature and a correct signature, an attacker can reveal the secret key. The authors only presented the theory, but yet demonstrated the threat that hardware faults might pose to cryptography. Today, publications on both side channel and fault attacks include ideas and describe practical implementations.

Thus, even if the mathematical principles of a certain system are strong enough to protect the system against purely mathematical cryptanalysis, it still might be insecure due to strong implementation attacks. Even if there are no mathematical weaknesses, side channel attacks or fault attacks might break such a system. Consequently, not only does sensitive information have to be secured with mathematically secure algorithms, but the concrete implementations of these algorithms also have to withstand physical attacks.

1.1 Thesis Statement

Today, it is generally agreed that side channel attacks and fault attacks pose a threat to cryptographic devices and to the secrets that they store. Accordingly, a great deal of research is conducted to find out which side channels can be used [77, 138] and how they can be optimally exploited [91]. Different algorithms are analyzed with respect to their susceptibility towards these attacks [33, 99, 141]. Findings about

[1] Sometimes, the term side channel attack is also used as broader term, e.g., [33, 69, 132]. In that terminology, passive side channel attacks are what we understand as side channel attacks, and active side channel attacks are what we understand as fault attacks. In this work, however, we do not use this terminology.

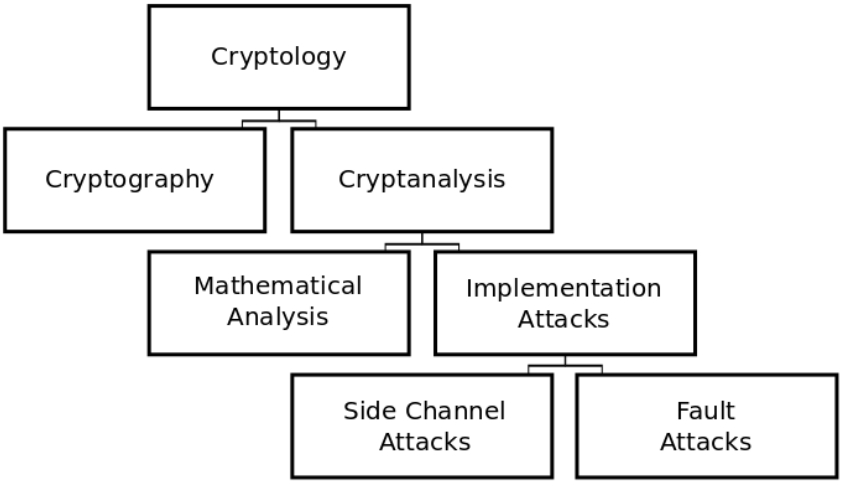


Figure 1.1: The role of implementation attacks in the field of cryptology.

side channel attack vectors lead to adjustments of the implementations [53] and countermeasures to prevent fault attacks are explored [35, 143]. More sophisticated attacks aim at breaking the improved implementations [10, 69, 106]. The knowledge about side channel attacks has even resulted in a new field of research with the goal to design cryptographic protocols that remain secure even in the presence of leakage from broad classes of side channels [14].

1.1.1 Problem Statement

Knowledge about side channel attacks and fault attacks influences the development of secure devices such as smartcards. Secure hardware vendors put a lot of effort into the development of devices secured against these kinds of attacks. Cryptographic algorithms are not implemented naively, but their implementations are meticulously designed around the knowledge gained from existing attacks.

Cryptographic devices and implementations should be secured against all realistically conceivable kinds of attacks - but it is difficult to determine which attacks are infeasible and which are not. Attacks which have already been conducted are taken into account, but there are many potential attack vectors which could be exploited and have not been used yet. In the past these attacks were often considered infeasible because they were believed to be hugely complex and expensive to perform. Thus, implementations lacked the necessary countermeasures to mitigate such attacks. Unfortunately, when the estimation of the real threat proved wrong afterwards, unprotected devices were in daily use for security applications. The development cycle of cryptographic devices and security applications is too long to react spontaneously to such developments. Vulnerable devices often remain in the field long after they are vulnerable to novel classes of analysis techniques.

Interestingly, in another field of cryptology, the research community does adapt

to future threats, although it is yet to be determined whether or not this threat will actually materialize. Quantum computers can invert certain one-way functions upon which the security of several modern cryptosystems relies [158]. They can solve both the integer factorization problem and the discrete logarithm problem, and thereby destroy the public-key scheme RSA, the Digital Signature Algorithm (DSA), and the Elliptic Curve Digital Signature Algorithm (ECDSA) [23]. Consequently, post-quantum cryptography is already a vibrant field of research, which aims at finding cryptographic schemes which will persist in the presence of quantum computers.

This thesis demonstrates that anticipation of future threats only affects the mathematical strength of cryptographic algorithms, but not their resistance to potential implementation attacks. Since the potential threat arising from quantum computers by far exceeds the one from yet another implementation attack, the comparison is not ideal. However, overestimating physical attack complexity is easy to mitigate by implementing more countermeasures against implementation attacks, have they already been realized or not.

1.1.2 Thesis Contributions

We present a side channel attack and a fault attack, both of which were considered unrealistic due to their physical complexity prior to the results demonstrated in this work. These attacks target established and time-proven algorithms as well as novel cryptographic protocols.

First, we present the photonic side channel. This side channel exploits highly spatially resolved photonic emission of the Device Under Attack (DUA) to reveal the secret key of a cryptographic algorithm. It was first presented in 2008, but was not considered a realistic threat due to the immense cost of more than 2,000,000 € for the measurement setup. When we presented a low-cost approach in 2012, we showed that the initial skepticism towards the applicability of this side channel was not justified. Based on this low-cost setup, we developed the theory of Simple Photonic Emission Analysis (SPEA) and Differential Photonic Emission Analysis (DPEA) and conducted practical attacks. Given the low-cost system and the methodology of SPEA and DPEA, the photonic side channel complements the cryptanalytic tools for attacking cryptography.

The second example is a higher-order fault attack against pairing computations. Pairings are the mathematical building blocks of Identity-Based Cryptography (IBC). Ever since they were suggested for this purpose, fault attacks against them were proposed. However, all of these attacks were only described theoretically until we published our results. Not only higher-order attacks, but even single-fault attacks against pairing computations were previously not practically realized. Second-order fault attacks were even considered to be an unrealistic attack scenario [180]. We conducted the first practical fault attack against a pairing computation, and even conducted it against a real-world pairing implementation. We successfully conducted a second-order fault attack against an implementation of the eta pairing from the RELIC toolkit [12], which was also used for the implementation of Pairing-Based

Cryptography (PBC) in Wireless Sensor Networks (WSNs) [127].

By presenting these two examples, we show that reliance upon physical attack complexity is not recommended when it comes to cryptography and the protection of sensitive information. In mathematics, proven results will always remain true. The human estimation of physical attack complexity, however, is error-prone. We have to face attackers who are better than we expect, and thus, cryptography needs also be secured against presumably physically infeasible attacks.

1.2 Structure of the Thesis

In Chapter 2, we give necessary background information. We provide background on elliptic curves and bilinear pairings and explain the AES algorithm and Identity-Based Encryption (IBE). We present relevant information on side channel attacks and on fault attacks. Then, we present the photonic side channel in Chapters 3 and 4. These chapters are based on [104, 105, 149, 150]. We explain the physics of photonic emission and the setups that we used for our photonic side channel attacks in Chapter 3. In Chapter 4, we present these attacks: we start with the Simple Photonic Emission Analysis and explain its application on all variants of AES. We also sketch attacks for other algorithms and discuss countermeasures. The chapter continues with Differential Photonic Emission Analysis. We present our results on AES for different distinguishers and also discuss countermeasures. The second example of an implementation attack which was assumed to be unrealistic is the second-order fault attack on pairing-based cryptography. We present our results in Chapter 5. First, we describe the general attack setup that we developed. Then, we explain how we utilized this setup to conduct the attack against the eta pairing. We used the free software implementation of the eta pairing from the RELIC toolkit [12]. Finally, we present the cryptanalytic steps from the faulty results to the secret key. This chapter is based on [31]. Our ideas for future work for the photonic side channel and for fault attacks against PBC are presented in Chapter 6. In Chapter 7, we conclude this thesis.

Chapter 2

Mathematical and Cryptological Background

First, we provide the necessary mathematical background information on elliptic curves and bilinear pairings. We assume the reader is familiar with abstract algebra and basic probability theory. Next, we present the AES algorithm and introduce Identity-Based Encryption (IBE). The chapter continues with the explanation of side channel attacks and fault attacks. The necessary terminology is presented and related work is discussed.

2.1 Elliptic Curves and Bilinear Pairings

Some of the definitions in this section can also be given more general. However, we wrote this background as concrete as possible. Therefore, the definitions are often customized to our needs and not as universally valid as they are in most text books.

2.1.1 Elliptic Curves

Definition 2.1 (Elliptic Curve)**.** *An elliptic curve E over a field K is the set of all points $P = (x_P, y_P)$ which satisfy the Weierstrass equation*

$$E : y^2 + a_1 x \cdot y + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6, \tag{2.1}$$

together with $\mathcal{O}$, the point at infinity. The coefficients $a_1, a_2, a_3, a_4, a_6 \in K$ are chosen so that the curve is nonsingular, i.e., for all points $P = (x_P, y_P)$ its partial derivatives do not vanish simultaneously.

An additive group structure can be defined on E. The group $(E, \oplus)$, which we refer to as only E, consists of all points satisfying the Weierstrass equation together with the point at infinity $\mathcal{O}$, which is the identity element in E. We denote the additive inverse of $P \in E$ with $-P$.

For any extension field L of K, we define the set of L-rational points on E as

$$\begin{aligned} E(L) = \{&(x, y) \in L \times L : \\ &y^2 + a_1 x \cdot y + a_3 y - x^3 - a_2 x^2 - a_4 x - a_6 = 0\} \cup \{\mathcal{O}\}. \end{aligned} \tag{2.2}$$

We denote the group order with $\#E$. The main result about the group order is Hasse's theorem.

Theorem 2.1 (Hasse's theorem)**.** *Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$. Then, $\#E(\mathbb{F}_q)$ satisfies*

$$|\#E(\mathbb{F}_q) - q - 1| \leq 2\sqrt{q}. \tag{2.3}$$

Proof. See [159, p. 138]. □

Hasse's theorem only defines boundaries for the cardinality of E. The exact determination of $\#E(\mathbb{F}_q)$, which is also called point counting problem, is of critical importance for many cryptographic applications. Methods to solve this problem are discussed in [29].

The following explanation of addition and scalar multiplication in E is based on the chord and tangent law [71], since the pairing algorithms deployed in this work make use of this construction.

Figure 2.1 shows how the sum $P \oplus Q$ of two distinct points $P, Q \in E$ can be determined. For two points P and Q, the line through P and Q is denoted with $l_{P,Q}$. Since the Weierstrass equation is of degree 3 in x, there will always be a third intersection point of $l_{P,Q}$ and the curve [159]. Assuming that $l_{P,Q}$ is neither the tangent line at P or Q nor vertical, we denote the third point of intersection with $-(P \oplus Q)$. If we mirror this point at the x-axis, we obtain $P \oplus Q$, the sum of P and Q. We can identify this point by drawing the vertical line $v_{-(P\oplus Q)}$ through $-(P \oplus Q)$ and then take the intersection of this line with the curve, see Figure 2.1. This point is the result $P \oplus Q$, the sum of P and Q. Hence, $v_{-(P\oplus Q)} = v_{P\oplus Q}$. In the case that $l_{P,Q}$ is the tangent of P or Q, the third point of intersection will be P or Q, respectively. In the case that $l_{P,Q}$ is vertical, i.e., Q is the reflection of P at the horizontal axis, the third point of intersection is $\mathcal{O}$.

To define scalar multiplication, we start with the definition of point doubling. Point doubling is the addition of the point to itself and thus, the procedure is analogous to point addition for two distinct points. To double P and compute $[2]P$, we draw the tangent line g_P through P at E. Since this tangent line intersects E with multiplicity 2 at P, there is only one further intersection point of g_P at E. In the case that the tangent line is not vertical, we denote this third point with $-(P \oplus P)$ and again reflect it at the x-axis to determine the point $P \oplus P$. The line connecting $-(P \oplus P)$ and $P \oplus P$ is again denoted with $v_{P\oplus P} = v_{-(P\oplus P)}$. In the case that the tangent line at P is vertical, the third point of intersection at E is $\mathcal{O}$, the point at infinity. Hence, we have $[2]P = \mathcal{O}$ in this situation. For the curve which is shown in Figure 2.1, there are three points with a vertical tangent line. These are $(0, 0)$, $(\sqrt{2}, 0)$, and $(-\sqrt{2}, 0)$.

Scalar multiplication of elliptic curve points can now be calculated with successive point additions and point doublings. With $[n]P$, we denote scalar multiplication of P with $n \in \mathbb{Z}$. For $P \in E$ and $n \in \mathbb{N}$, we have

$$[n]P = \overbrace{P \oplus P \ldots \oplus P}^{n \text{ times}}. \tag{2.4}$$

Consequently, we have

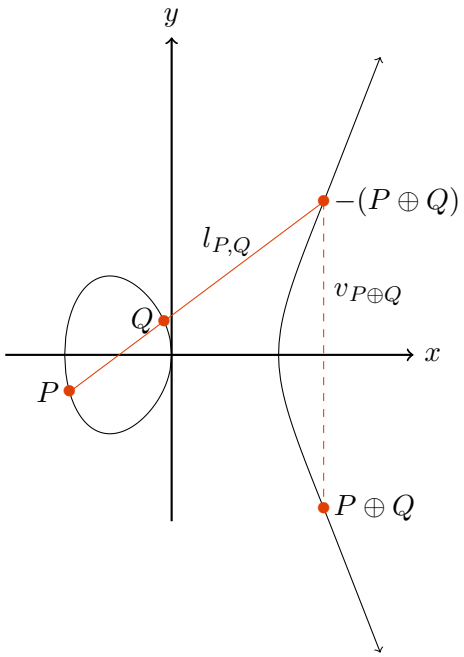


Figure 2.1: Point addition of two distinct points on the elliptic curve $y^2 = x^3 - 2x$.

$$[-n]P = \overbrace{-P \oplus -P \ldots \oplus -P}^{n \text{ times}} \tag{2.5}$$

and

$$[0]P = \mathcal{O}. \tag{2.6}$$

The point representation that is used in this work is referred to as affine coordinate system and affine coordinates. To speed up group operations and to mitigate side channel attacks, points on elliptic curves can also be represented in different coordinate systems such as projective coordinates [87] and Jacobian coordinates [120].

Definition 2.2 (Torsion Point)**.** *Let E be an elliptic curve. For $r \in \mathbb{Z}$ we define the set*

$$E[r] = \{P \in E : [r]P = \mathcal{O}\}. \tag{2.7}$$

These points of finite order are called torsion points and $E[r]$ is the group of r-torsion points.

Definition 2.3 (Supersingular Curve)**.** *Let E be an elliptic curve defined over a field with characteristic p. If $E[p^n] = \{\mathcal{O}\}$ for all $n \geq 1$, E is called supersingular. Otherwise, E is called ordinary.*

Definition 2.4 (Distortion Map)**.** *Let E be a supersingular elliptic curve and $r \in \mathbb{N}$ with $r \geq 2$. Let $P \in E$ so that $P \in E[r]$ and P is of exact order r. We call a homomorphism $\psi : E \to E$ distortion map, if $\{P, \psi(P)\}$ is a basis for $E[r]$ and hence, $\psi(P)$ is not in the cyclic subgroup generated by P.*

Distortion maps were introduced in [177].

Definition 2.5 (Twist)**.** *Let E and E' be elliptic curves over $\mathbb{F}_q$. If there is an isomorphism $\phi_d : E' \to E$ defined over $\mathbb{F}_{q^d}$ with minimal d, then E' is a degree-d twist of E.*

We refer the reader to [29, 30, 52, 101, 159] for a thorough treatment of elliptic curves. Information about elliptic curve cryptography can be found in [29, 30, 52, 87].

Elliptic Curve Discrete Logarithm

The Discrete Logarithm Problem (DLP) is a well-known mathematical problem on which cryptographic algorithms such as ElGamal encryption, the ElGamal signature scheme, and the Diffie-Hellman key establishment are based [115]. The analog problem exists for elliptic curves. It is called Elliptic Curve Discrete Logarithm Problem (ECDLP).

Definition 2.6 (Elliptic Curve Discrete Logarithm Problem)**.** *Let E be an elliptic curve over a finite field $\mathbb{F}_q$ and $P, Q \in E$ so that there is $n \in \mathbb{Z}$ with $Q = [n]P$. The Elliptic Curve Discrete Logarithm Problem (ECDLP) is, given P and Q, to find n.*

As it is the case for the DLP on natural numbers, there are simple instances of the ECDLP, while the problem is not efficiently solvable for other groups and points [29]. For details on elliptic curve discrete logarithms and their application, we refer the reader to [71].

2.1.2 Bilinear Pairings

Bilinear pairings are bilinear maps defined over groups on elliptic curves. Originally, they have been used for cryptanalytic techniques [116]. In 2001, however, they gained the research community's attention when they were used to realize IBE [38], see Section 2.2.2. Today, a wide range of different pairings is used [11] and several cryptographic protocols such as attribute-based encryption [144], identity-based signatures [89], and key agreement protocols [94] are based on pairings. Moreover, pairings help to secure useful technologies such as WSNs [129]. Ever since pairings were proposed to be used for IBE, cryptanalysis of pairings and pairing-based schemes became an active field of research, e.g., [7, 72, 183].

For the definition of pairings, we first have to define the embedding degree.

Definition 2.7 (Embedding Degree)**.** *Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$ and let r be an integer coprime to q with $r \mid \#E(\mathbb{F}_q)$. The embedding degree (with respect to r) is the smallest natural number k such that $r \mid (q^k - 1)$. It satisfies $\mathbb{F}_{q^k} = \mathbb{F}_q(\mu_r)$, where μ_r denotes the group of rth roots of unity in $\overline{\mathbb{F}}_q$.*

Thus, the embedding degree is the smallest natural number so that $\mathbb{F}_{q^k}^*$ has a subgroup of order r, i.e., $r \mid (q^k - 1)$. Since k is the order of q modulo r, i.e., k is the order of q in the unit group $\mathbb{Z}_r^*$, it follows that k divides Euler's totient function $\phi(r)$. Hence, if r is prime, then k divides $(r-1)$.

Definition 2.8 (Pairing)**.** *Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$. We define three finite abelian groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. The groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of E, while $\mathbb{G}_T$ is a subgroup of $\mathbb{F}_{q^k}^*$, with k the embedding degree. A pairing is an efficiently computable, non-degenerate bilinear map*

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T. \tag{2.8}$$

We refer the reader to [52] for a thorough treatment of bilinear pairings and their application. For details on the selection of pairing-friendly curves we refer to [70].

Most pairings $e(P, Q)$ on elliptic curves are computed by first computing the so-called Miller function $f_{n,P}(Q)$ [121] followed by an exponentiation to the power $z = (q^k - 1)/r$. To understand this Miller function and to study pairings in more detail, we have to address divisors and some of their characteristics.

Definition 2.9 (Divisor)**.** *For an elliptic curve E, a divisor D is a formal sum*

$$D = \sum_{P \in E} n_P \cdot \langle P \rangle. \tag{2.9}$$

Only finitely many of the coefficients $n_P \in \mathbb{Z}$ are nonzero. The divisor associated to the point $P \in E$ is denoted with $\langle P \rangle$.

The divisors generated by the points of E form a free abelian group. We denote the set of all divisors generated by the points of E with Div(E).

We want to define the divisor associated to a rational function. For this definition, we first need to define the order of a rational function at a point of an elliptic curve [159].

Definition 2.10 (Order of a function at a point)**.** *Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$, $P \in E$, and $f \in \overline{\mathbb{F}_q}[E]_P$. The order of f at P is*

$$\mathrm{ord}_P(f) = \sup\{d \in \mathbb{Z} : f \in M_P^d\}, \tag{2.10}$$

where M_P denotes the maximal ideal of the local ring $\overline{\mathbb{F}_q}[E]_P$.

For a nonzero rational function $f = g/h \in \overline{\mathbb{F}_q}(E)$ with $g, h \in \overline{\mathbb{F}_q}[E]_P$, we define

$$\mathrm{ord}_P(f) = \mathrm{ord}_P(g) - \mathrm{ord}_P(h). \tag{2.11}$$

If $\mathrm{ord}_P(f) > 0$, we say that f has a zero at P while we say that f has a pole at P if $\mathrm{ord}_P(f) < 0$. If $\mathrm{ord}_P(f) \geq 0$, then f is defined at P and $f(P)$ can be evaluated. Otherwise f has a pole at P and we write $f(P) = \infty$.

Definition 2.11 (Divisor associated to a function)**.** *Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$, and $f \in \overline{\mathbb{F}_q}(E)$ a nonzero rational function. The divisor associated to f is*

$$\mathrm{div(f)} = \sum_{\mathrm{P} \in \mathrm{E}} \mathrm{ord_P(f)} \cdot \langle \mathrm{P} \rangle. \tag{2.12}$$

A divisor is called principal if it is equal to a divisor that is associated to a function. Thus, $D \in \mathrm{Div(E)}$ is principal if $D = \mathrm{div(f)}$ for some nonzero rational function $f \in \overline{\mathbb{F}_q}(E)$. We say that two divisors $D_1, D_2 \in \mathrm{Div(E)}$ are linearly equivalent if $D_1 - D_2$ is principal. To denote that two divisors are linearly equivalent, we write $D_1 \sim D_2$.

Definition 2.12 (Support of a Divisor)**.** *Let E be an elliptic curve and $D \in \mathrm{Div(E)}$ with $D = \sum_{P \in E} n_P \cdot \langle P \rangle$. The support of D is the set of all points $P \in E$ with $n_P \neq 0$. Two divisors are coprime when their supports are disjoint.*

For a divisor $D \in \mathrm{Div(E)}$ and a principal divisor div(f) for some nonzero rational function $f \in \overline{\mathbb{F}_q}(E)$, we can define the evaluation of f at D if D and div(f) are coprime. Then, we define $f(D) = \prod_{P \in E} f(P)^{n_P}$.

With the help of divisors, we can now define the Miller function, which is the heart of most pairings.

Definition 2.13 (Miller function)**.** *Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$. A rational function $f_{n,P} \in \mathbb{F}_q(E)$, with $P \in E$ and $n \in \mathbb{N}$, is a Miller function if its divisor satisfies*

$$\mathrm{div(f_{n,P})} = \mathrm{n} \cdot \langle \mathrm{P} \rangle - \langle [\mathrm{n}]\mathrm{P} \rangle - (\mathrm{n}-1)\langle \mathcal{O} \rangle. \tag{2.13}$$

The Miller function can also be defined recursively as follows [34, 121]:

$$f_{n+1,P} := \begin{cases} 1 & \text{if } n = 0 \\ f_{n,P} \cdot l_{P,[n]P} / v_{[n+1]P} & \text{else.} \end{cases}$$

As stated above, most pairings on elliptic curves are computed by first evaluating the Miller function, followed by an exponentiation. The evaluation of the Miller function can be efficiently computed with the Miller Algorithm, see Algorithm 2.1.

The Miller Algorithm successively computes the required function in $\lfloor \log_2(n) \rfloor$ iterations. Since this algorithm utilizes a `for` loop, see Lines 2 to 9, this evaluation is called Miller loop [44]. The value n, which determines the number of loop iterations, is often called Miller bound. The variable f, which is updated during the computation of the `for` loop until it stores the value of the evaluation of the Miller function, is called Miller variable.

Independent of the concrete design of a pairing, there are three different types of pairings. Following [73], we distinguish between pairings of Type 1, Type 2, and Type 3. These three basic types differ in the similarity of their domains $\mathbb{G}_1$ and $\mathbb{G}_2$.

- In a Type 1 pairing, both groups are equal, i.e., $\mathbb{G}_1 = \mathbb{G}_2$.

Algorithm 2.1 Miller Algorithm and final exponentiation.

Require: $P, Q \in E$, $n = \sum_{j=0}^{t-1} n_j 2^j$ with $n_j \in \{0, 1\}$ and $n_{t-1} = 1$
Ensure: $f_{n,P}(Q)^z$

1: $T \leftarrow P$, $f \leftarrow 1$
2: **for** $j = t - 2 \ldots 0$ **do**
3: $\quad f \leftarrow f^2 \cdot g_T(Q)/v_{[2]T}(Q)$
4: $\quad T \leftarrow [2]T$
5: $\quad$ **if** $n_j = 1$ **then**
6: $\quad\quad f \leftarrow f \cdot l_{T,P}(Q)/v_{T \oplus P}(Q)$
7: $\quad\quad T \leftarrow T \oplus P$
8: $\quad$ **end if**
9: **end for**
10: $f \leftarrow f^z$ ▷ final exponentiation
11: **return** f

- In a Type 2 pairing, the groups are different, i.e., $\mathbb{G}_1 \neq \mathbb{G}_2$, but it exists an efficiently computable homomorphism $\phi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$.
- In a Type 3 pairing, the groups are different, i.e., $\mathbb{G}_1 \neq \mathbb{G}_2$, and there are no efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$.

Type 1 pairings are also called symmetric pairings, while Type 2 and Type 3 pairings are called asymmetric pairings. The choice of the type of the pairing is related to the elliptic curve that is used, e.g., Type 1 pairings are generally implemented using supersingular curves over $\mathbb{F}_p$, $\mathbb{F}_{2^m}$, and $\mathbb{F}_{3^m}$ [153].

Those pairings which are used in cryptography require that their inversion is not efficiently computable. There are four mathematical problems connected to pairing inversion.

Definition 2.14 (FAPI-1)**.** *Given a point $P \in \mathbb{G}_1$ and a value $\alpha \in \mathbb{G}_T$, both chosen at random, the fixed argument pairing inversion problem (FAPI-1) is to find $Q \in \mathbb{G}_2$ such that $e(P, Q) = \alpha$ [72].*

The problem can also be defined with P unknown and $Q \in \mathbb{G}_2$ chosen at random. The problem is then called FAPI-2.

Both FAPI-1 and FAPI-2 treat the pairing as a black box and only consider the inputs and the output. However, analogous to the two steps of the pairing calculation, i.e., Miller Algorithm and final exponentiation, the pairing inversion can also be treated as a two-step process [97]. Hence, FAPI-1 is usually split into two parts in the literature: the exponentiation inversion [44] and the Miller inversion [72].

Definition 2.15 (Exponentiation Inversion)**.** *Given the output of the pairing as well as $P \in \mathbb{G}_1$ and the final exponent z, the exponentiation inversion problem is to find the correct preimage of the final exponentiation, i.e., the field element $f_{n,P}(Q)$.*

Definition 2.16 (Miller Inversion)**.** *The Miller inversion problem is, given n, $P \in \mathbb{G}_1$, and a field element $f_{n,P}(Q)$, to find the correct input $Q \in \mathbb{G}_2$.*

Recently, a novel theoretic idea for pairing inversion was presented [97]. Kanayama and Okamoto show that the Miller inversion does not have to be solved in several cases provided that a generic algorithm for solving the exponentiation inversion exists. Thus, given such an algorithm, pairing inversion can be reduced to exponentiation inversion for a special family of pairings, the Ate$_i$ pairings. This work was later revised and extended [44].

Related to the inversion of pairings, especially when it comes to fault attacks, is the Hidden Root Problem (HRP), which was introduced only in 2008 [175]. We define the HRP exactly as it was defined in the original publication.

Definition 2.17 (Hidden Root Problem)**.** *Let $\mathbb{F}_q$ be a finite field with $q = p^n$ elements, where p is prime and let e be a positive integer with $e \mid (q-1)$. Let $f_x(\cdot) : \mathcal{D}(\mathbb{F}_q) \to \mathbb{F}_q$ be a specified map, i.e., the precise description of which is given, depending on x from a domain $\mathcal{D}$ to $\mathbb{F}_q$. Let $\mathcal{O}_x(\cdot)$ denote an oracle that on input $\alpha \in \mathcal{D}$ returns*

$$\xi_\alpha = f_x(\alpha)^e \tag{2.14}$$

for a fixed secret $x \in \mathbb{F}_q$. The Hidden Root Problem is to recover x in expected polynomial time in $\log q$ by querying the oracle repeatedly for chosen $\alpha_i \in \mathcal{D}(\mathbb{F}_q)$.

The Tate Pairing

We describe the Tate pairing and its reduced variant following [176].

Definition 2.18 (Tate Pairing)**.** *Let E be an elliptic curve defined over $\mathbb{F}_q$. Let $r \mid \#E(\mathbb{F}_q)$ so that r and q are coprime. Let k be the embedding degree. The Tate pairing $\hat{t}$ is defined as*

$$\begin{aligned} \hat{t} : E\left(\mathbb{F}_{q^k}\right)[r] \times E\left(\mathbb{F}_{q^k}\right)/rE\left(\mathbb{F}_{q^k}\right) &\to \mathbb{F}_{q^k}^*/\left(\mathbb{F}_{q^k}^*\right)^r \\ (P,Q) &\mapsto \hat{t}(P,Q) = f_{r,P}(D_Q). \end{aligned} \tag{2.15}$$

Here, $\mathrm{div}(\mathrm{f_{r,P}}) = \mathrm{r}\cdot\langle \mathrm{P}\rangle - \mathrm{r}\cdot\langle\mathcal{O}\rangle$ *and $D_Q \sim \langle Q\rangle - \langle\mathcal{O}\rangle$ is coprime to* $\mathrm{div}(\mathrm{f_{r,P}})$.

To yield a unique result instead of a representative of an equivalence class, we define the reduced Tate pairing t.

Definition 2.19 (Reduced Tate Pairing)**.** *Let E be an elliptic curve defined over $\mathbb{F}_q$. Let $r \mid \#E(\mathbb{F}_q)$ so that r and q are coprime. Let k be the embedding degree. The reduced Tate pairing t is defined as*

$$\begin{aligned} t : E\left(\mathbb{F}_{q^k}\right)[r] \times E\left(\mathbb{F}_{q^k}\right)/rE\left(\mathbb{F}_{q^k}\right) &\to \mu_r \subset \mathbb{F}_{q^k}^* \\ (P,Q) &\mapsto t(P,Q) = \hat{t}(P,Q)^{(q^k-1)/r}. \end{aligned} \tag{2.16}$$

Thus, the reduced Tate pairing consists of two stages: first the Miller function that is parameterized by P is evaluated at D_Q, then a final exponentiation follows to ensure that the algorithm outputs a unique value.

The eta Pairing

The eta pairing can be regarded as an optimized version of the reduced Tate pairing [19, 176]. The optimization consists in a shortened Miller loop. To define the eta pairing, we first need to introduce the Frobenius endomorphism.

Definition 2.20 (Frobenius Endomorphism)**.** *Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$. The endomorphism*

$$\begin{aligned} \phi_q : E &\to E, \\ (x, y) &\mapsto (x^q, y^q) \end{aligned} \tag{2.17}$$

is called Frobenius endomorphism. The group $E(\mathbb{F}_q)$ of $\mathbb{F}_q$-rational points on E is fixed by ϕ_q since the qth power is the identity on $\mathbb{F}_q$.

Definition 2.21 (eta Pairing)**.** *Let E be a supersingular elliptic curve defined over $\mathbb{F}_q$. Let $r \mid \#E(\mathbb{F}_q)$ so that r and q are coprime and let k be the embedding degree. Let n be any integer with $n = q \mod r$ so that $(n^k - 1)/r$ and r are coprime. Let $\mathbb{G}_1, \mathbb{G}_2 \subseteq E$ be restricted to the Eigenspaces of Frobenius with $\mathbb{G}_1 = E[r] \cap \ker(\phi_q - [1])$ and $\mathbb{G}_2 = E[r] \cap \ker(\phi_q - [q])$. The eta pairing η_n is defined as*

$$\begin{aligned} \eta_n : \mathbb{G}_1 \times \mathbb{G}_2 &\to \mu_r \subset \mathbb{F}_{q^k}^* \\ (P, Q) &\mapsto \eta_n(P, Q) = f_{n,P}(Q)^{(q^k-1)/r}. \end{aligned} \tag{2.18}$$

2.2 Cryptographic Algorithms and Protocols

This section first presents the AES algorithm, which is the target of the photonic side channel attacks from Chapter 4. Then, background information on IBC is given.

2.2.1 The Advanced Encryption Standard

The analyses in Chapter 4 focus on the Advanced Encryption Standard (AES). AES is a symmetric encryption algorithm based on the Rijndael cipher [58]. It is ratified as a standard by the National Institute of Standards and Technology of the USA. AES has a fixed block size of 128 input bits and operates on a 4×4 matrix of bytes, named the state. Depending on the length of the key, which is 128, 192, or 256 bits, the cipher is termed AES-128, AES-192, or AES-256. The algorithm is specified as a number of rounds that transform the input plaintext into the ciphertext. AES consists of 10, 12, and 14 rounds for 128-, 192-, and 256-bit keys, respectively. Each round consists of four operations `SubBytes`, `ShiftRows`, `MixColumns`, `AddRoundKey`, except for the final round, which skips the `MixColumns` operation. Additionally, there is an initial `AddRoundKey` operation before the first round. Algorithm 2.2 shows the sequence of the rounds for AES-128.

Regarding AES-128, the initial `AddRoundKey` operation uses the complete secret 128-bit key. Then, each 128-bit round key k_i is derived deterministically from

Algorithm 2.2 AES-128 Algorithm.

Require: plaintext $m \in \{0,1\}^{128}$, key $k \in \{0,1\}^{128}$
Ensure: ciphertext $c \in \{0,1\}^{128}$

1: state $s \leftarrow m$
2: $s \leftarrow \text{AddRoundKey}(s, k)$
3: **for** i = 1 ... 9 **do**
4: $s \leftarrow \text{SubBytes}(s)$
5: $s \leftarrow \text{ShiftRows}(s)$
6: $s \leftarrow \text{MixColumns}(s)$
7: $s \leftarrow \text{AddRoundKey}(s, k_i)$
8: **end for**
9: $s \leftarrow \text{SubBytes}(s)$
10: $s \leftarrow \text{ShiftRows}(s)$
11: $s \leftarrow \text{AddRoundKey}(s, k_{10})$
12: $c \leftarrow s$
13: **return** c

this original secret key with Rijndael's key schedule [58]. During each round of AES-192 and AES-256, also a 128-bit round key is used. The key for the initial AddRoundKey operation consists of the first 128 bits of the secret 192- and 256-bit secret key, respectively. The round key of the first round consists of the remaining bits of the secret key. Regarding AES-192, the second half of this round key is derived with the key schedule, while for AES-256, the key schedule is used only from the second round.

In the AddRoundKey step, each byte of the state is combined with a byte of the round key using the exclusive or operation ($\oplus$). In the SubBytes step, each byte of the state is substituted by an affine transformation of its multiplicative inverse over the Galois field $\mathbb{F}_{2^8}$. This is the only operation that provides non-linearity in the algorithm. Since this deterministic operation is very costly, a precomputed 8-bit lookup table is often used. This so-called S-Box is shown in Table 2.1. The substitution value of a byte $b = b_7|b_6|b_5|b_4|b_3|b_2|b_1|b_0$ with $b_i \in \{0,1\}$ for all $i \in \{0,\ldots,7\}$ is calculated by deriving the row and column numbers from the byte value. The row number consists of the value of the four Most Significant Bits (msb), i.e., b_7 to b_4, and the column number consists of the value of the four Least Significant Bits (lsb), i.e., b_3 to b_0. In the ShiftRows step, each row of the state matrix is shifted to the left by $0, 1, 2$ or 3 bytes. In the MixColumns step, the four bytes of each column of the state matrix are combined using an invertible linear transformation, resulting in another four bytes.

The memory access patterns of AES are particularly susceptible to cryptanalysis [22, 83, 130]. Optical emissions related to AES S-Box accesses are also exploited in the photonic side channel attacks in Chapter 4. The algorithm running on our Device Under Attack (DUA) consists of a software AES implementation. To increase

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Table 2.1: The AES S-Box, used during the SubBytes operation, in hexadecimal representation. The 4 msb of the input byte determine the row and the 4 lsb determine the column. The element accessed by the input is the substitution value and hence, the next state value.

the frequency of the execution, only the first AddRoundKey and SubBytes operations were computed on the chip after which the input was reset and the measurement restarted. Table 4.8 shows the assembly code of our SubBytes implementation.

2.2.2 Identity-Based Cryptography from Pairings

IBC was invented by Shamir, who presented the idea of this public-key system in 1984 [157]. However, he could not explain back then how to realize such encryption schemes mathematically. Nearly two decades later, in 2001, Boneh and Franklin showed that elliptic curves and pairings can be used to realize IBE [38].

IBE is a form of asymmetric cryptography where the identity of a user is at once his public key. In any form of communication and cryptography, the sender of a message has to know an information about the receiver which is uniquely assigned to him, such as his email address. The idea of IBE is that this information is sufficient to use cryptography and that no further information such as a dedicated public key is necessary. Therefore, IBE facilitates especially those systems which have to manage large numbers of key material.

A system which implements IBE needs a trusted third party or trusted authority, the so-called Private Key Generator (PKG). Following Boneh and Franklin's initial scheme, four probabilistic polynomial time algorithms are used:

- Setup: This algorithm generates all system parameters and the secret master key. Only the PKG knows this master key. The public system parameters include descriptions of the plaintext space $\mathcal{P}$ and the ciphertext space $\mathcal{C}$ and of the set of possible identities.

- Extract: The private key of a user is extracted from the secret master key and his identity, i.e., his public key.

- Encrypt: The randomized encryption of a message $m \in \mathcal{P}$ is computed based on the public system parameters and the identity of the receiver, i.e., his public key.

- Decrypt: The deterministic decryption function outputs a plaintext for given ciphertext $c \in \mathcal{C}$ and the receiver's private key.

In the FullIdent scheme proposed by Boneh and Franklin, both in the Encrypt and Decrypt step a bilinear pairing is computed [38]. During the Decrypt step, the input to the pairing consists of a part of the ciphertext and the secret key of the receiver of that ciphertext. Hence, it has to be ensured that an attacker cannot gain any information about the secret input to a pairing. Attacks on pairings have therefore become a prominent strand within the research on pairings.

An advantage of IBE is the facilitated key management. Certificates are no longer necessary. Furthermore, the sender can send an encrypted message to the receiver even if the receiver does not know his private key yet. It is likewise possible to provide public keys with a validity period which results in a simple method for key revocation. On the other hand, key escrow is immanent in IBE systems since each private key can be derived from the master key at any time. This is an advantage in the case that each user really trusts the PKG. Otherwise, it is a drawback of such systems. In the case of identity-based signatures, the nonexisting non-repudiation is another disadvantage for the same reason.

Identity-Based Cryptography for Wireless Sensor Networks IBE is especially well suited for those devices with a constrained power supply, such as smartcards and WSNs. On the one hand, WSNs benefit from the non-necessary costly verification of certificates. On the other hand, it is easy to add an additional node since there are no certificates of that node to be verified [48]. Hence, IBE can be used to solve the key distribution problem in WSNs [127]. Another decided advantage is that the existence of a PKG is immanent to WSNs by means of the base station which connects the WSN to other networks. In addition to these advantages, the computation of pairings necessary for IBE is for a given security level more efficient than classical public key cryptography [129]. However, since the PKG can derive all private keys from the master key, in general it is considered a caveat that all users of an IBE system have to trust the PKG.

2.3 Side Channel Attacks

In a cryptographic side channel attack, the attacker captures physical characteristics of the DUA while it performs computations with secret data. She analyzes these characteristics to reveal secret information, such as the secret key of an encryption algorithm. Side channel attacks have been a significant research area since the seminal papers of Kocher in 1996 and 1999, which introduced the timing [102] and the power side channel [103]. Since then, a plethora of other side channels, applications, and analysis methods have been presented. In this section, we provide the necessary background information about side channel attacks and present a selection of relevant work. Side channel attacks are also called passive or non-invasive attacks [33].

The attacker collects a number of so-called traces to reveal the secret information. In the case of power consumption, "a trace refers to a set of power consumption measurements taken across a cryptographic operation" [103]. Thus, each trace consists of a set of side channel leakage values, each related to a distinct point in time. Side channel attacks are divided into univariate and multivariate attacks. Univariate attacks exploit the leakage of only a single point in time [60], while multivariate attacks exploit multiple aspects of the measurements jointly. Side channel attacks combining multiple points of leakage are also called higher-order attacks [56, 103]. Since protected implementations can generally not be successfully attacked with a univariate attack, higher-order attacks are also a means to break countermeasures [95, 111].

Since in real-world applications access to the DUA is often restricted, an attacker cannot get an unlimited number of traces. Template attacks compensate for a smaller number of traces during the attack by adding an additional attack phase in which the attacker has unlimited access to an identical experimental device [45]. During this phase, she can execute arbitrary code on this device and thereby derive templates which model both the signal and the noise. These templates improve the efficiency of the attack in terms of the required number of traces and let the attacker fully utilize the leakage information from each trace.

For the analysis of side channel information, many different distinguishers exist [60, 90]. Distinguishers are the statistical methods which are applied to side channel measurements. However, often the choice of the distinguisher is of minor importance [113].

If a secret key is the target of a side channel attack, it is often not revealed as a whole, but in separate chunks. These separate chunks are called subkeys. If the AES algorithm is the target of the attack, for instance, each key byte is generally revealed independently.

The success rate of an attack can be measured based on its Global Success Rate (GSR) and its Partial Success Rate (PSR). The Global Success Rate is defined as the probability that the complete key is ranked first, i.e., it is the probability of getting the correct value for all key bytes simultaneously [85]. The Partial Success Rate is defined as the probability that the correct subkey is ranked first among all

possible subkeys, i.e., the Partial Success Rate (PSR) is the probability of obtaining the correct value, computed independently for each key byte [90]. Since each key byte has its own PSR, often the minimal PSR (min PSR) is used to describe the attack efficiency [165].

Example 2.1. *Assume that we have a set of traces and want to attack a 16-byte key. We randomly choose ten subsets of the traces and try to reveal the secret key for each of these subsets. In five cases, we reveal the secret key completely, while in the remaining 5 cases, we do not correctly identify the Least Significant Byte (LSB). Then, the Global Success Rate (GSR) is* $\frac{5}{10} = 0.5$*, while the PSR is* 1 *for the first 15 key bytes and* 0.5 *for the LSB. Thus, min PSR is* 0.5*. Assume now that we never reveal the secret key completely, but in experiment* i*,* $i \in \{1, \ldots, 10\}$*, all key bytes except for key byte* i *are revealed. Then, we have min PSR* $= 0.9$*, but GSR* $= 0$*.*

2.3.1 Timing Attacks

In 1996, a timing attack against several cryptographic algorithms was the first published side channel attack [102]. In a timing attack, the attacker analyzes the time required to perform operations which involve secret key material to extract information about that key material. Often, timing attacks target algorithms which employ an exponentiation with a secret exponent, since the timing strongly depends on the value of the processed exponent bits in unprotected implementations [146]. Timing attacks can also be conducted remotely, e.g., against network servers [42].

A variant of timing attacks, both local and remote, are cache attacks. In a cache attack, the timing information provides information about cache hits and misses and hence, about the cache entries. Therefore, cache attacks often target algorithms using table lookups such as AES. In 2004, Bernstein conducted a known-plaintext cache timing attack on the OpenSSL AES implementation that uses precomputed tables [22]. He extracted a complete 128-bit AES key. The mathematical analysis of our attack presented in Section 4.1 is similar to that analysis. In 2005, Percival revealed an OpenSSL 1024-bit RSA private key by exploiting simultaneous multithreading [135]. This OpenSSL implementation used RSA-CRT for private-key operations. During the attack, 310 out of 512 bits per exponent could be extracted, which is enough for factorizing the modulus.

2.3.2 Power Analysis

The second side channel that was exploited is power analysis. In 1999, Kocher et al. presented Simple Power Analysis (SPA) and Differential Power Analysis (DPA). They define SPA as "a technique that involves directly interpreting power consumption measurements collected during cryptographic operations". In contrast, DPA does not interpret the traces directly, but uses "statistical functions tailored to the target algorithm" [103]. In this seminal work, the Data Encryption Standard (DES) was attacked. After the publication of these results, countermeasures against power analysis have been developed, e.g., masking and hiding [112]. Masking means to

make the processed value uncorrelated to the algorithmic value, while hiding means to make the power consumption uncorrelated to the processed value. However, the analysis methods also evolved, and higher-order attacks against DPA-resistant software and hardware have been published [51, 117]. A few years ago it was even shown that successful SPAs are still possible, despite several implemented countermeasures like message padding [55, 106].

Power analysis attacks also target novel algorithms like PBC. In 2006, both SPA and DPA on the eta pairing over binary fields were presented [100]. The authors suggest randomization and blinding as defense against such attacks, both of which are already used in other algorithms such as RSA. In 2009, a DPA against pairing computations was simulated [64]. It was shown how the secret input point to the Miller Algorithm can be revealed by analyzing a modular multiplication and an addition. In 2013, these results were improved and it was theoretically described how the secret input to the Miller Algorithm can be revealed only by a DPA of a modular multiplication [33].

2.3.3 Electromagnetic Analysis

The analysis of electromagnetic radiation as a side channel was already mentioned in the seminal work on power analysis [103]. Electromagnetic emanation is a three-dimensional vector field which changes over time. Instead of observing the near-field emanation of the whole Integrated Circuit (IC), the observation can be restricted to a certain location such as a specific component of the IC. Such localized measurements allow for side channel attacks which exploit location-dependent information leakage, though to a lesser extent than is the case for Photonic Emission Analysis. The level of localization is related to the diameter of the magnetic coil usually used in electromagnetic side channel attacks to acquire the measurements. We refer to Heyszl's PhD thesis for more information on the strengths and limitations of high-resolution measurements for electromagnetic side channel attacks [92].

In 2001, two publications on the electromagnetic (EM) side channel appeared [75, 138]. Both publications stress the advantage that EM analysis has over power and timing analysis: exploitation of locally resolved data leakage. A technical description of EM attacks on smartcards is given in [138]. The authors explain the physics of EM radiation and describe how attacks can be practically realized. However, EM radiation is not analyzed for cryptanalytic purposes in this work. Gandolfi et al. showed by example of a Simple Electromagnetic Analysis (SEMA) against RSA and a Differential Electromagnetic Analysis (DEMA) against the DES how EM attacks can be practically conducted [75]. Both implementations were unprotected against side channel attacks. The authors state that in terms of their experiments, EM analysis outperforms power analysis. In 2002, Agrawal et al. presented a systematic investigation of the EM side channel for Complementary Metal-Oxide-Semiconductor (CMOS) devices [8]. They present concrete results for two different smartcards. They show that EM analysis can even be successful in the presence of power analysis countermeasures. An extended version of their work is also avail-

able [9]. In this work, especially higher order attacks and assessment methodologies for these are examined in more detail.

In 2012, location-dependent electromagnetic leakage was successfully exploited in an attack on an elliptic curve scalar multiplication implementation on a Field Programmable Gate Array (FPGA) using a near-field EM probe [93]. The authors scanned the die surface and collected EM traces at every point. They demonstrated that location-dependent leakage can be used in a template attack and countermeasures against system-wide leakage thus can be circumvented.

2.3.4 Other Side Channels

In addition to these classical side channels, there are also other sources of side channel leakage which can be exploited.

In 2008, a cold boot attack was presented [86]. The authors show how disc encryption keys which are stored in Dynamic Random-Access Memory (DRAM) can be easily stolen if the attacker has physical access to the computer. Contrary to many other side channel attacks, this work assumes a very realistic attack scenario and this attack is a serious security vulnerability. The attack becomes possible since data stored in DRAM does not vanish instantaneously, but fades away gradually when the power is turned off. By cooling the memory, this process can even be slowed down. This attack is considered to be a side channel attack even though the attackers do not capture physical characteristics of the DUA while it performs computations with secret data. They do, however, exploit the physical implementation of a cryptosystem.

In 2013, the acoustic side channel was presented [77]. The authors show how a 4096-bit RSA key can be extracted by analyzing the acoustic frequency spectrum during decryption. The decryption was performed by GnuPG running on a laptop, while the acoustic emanation was captured only with a plain mobile phone. Thus, this attack was launched with low-cost equipment. Although the acoustic side channel has a very low bandwidth which makes such attacks more difficult to conduct, specially crafted chosen messages lead to more discernible leakage in the presented attack.

2.4 Fault Attacks

The principle of fault attacks is to disturb the computation of cryptographic operations by induction of one or more faults. From the faulty result, the attacker learns information about the internal sensitive data such as the secret key. The first fault attack against a cryptographic algorithm was presented in 1997 [37]. Since then, fault attacks have been applied against various cryptographic algorithms [174] and became a standard tool to facilitate cryptanalysis. The attack assumptions can be described in detailed fault models, which include the location and the timing of the fault, and the number and kind of faults. Nowadays, many techniques exist to induce faults, e.g., clock glitching, power glitching, and laser beams [17]. To thwart

countermeasures against fault attacks, even two faults within one computation have been performed [99, 170]. These attacks are often called second-order attacks [61]. More generally, we call a fault attack with more than one fault a higher-order attack.

2.4.1 RSA

The first fault attack against a cryptographic algorithm, known as the Bellcore attack, was presented in 1997 by Boneh, DeMillo and Lipton against the RSA signature scheme [37]. They showed that the RSA modulus can be factorized if an attacker induces a fault into the computation of one of the two parts of the signature generation in case the RSA CRT version is used. With a single faulty signature and a correct signature under the same secret key, an attacker can reveal the secret key.

Since then, RSA has been a popular target for fault attacks. A hardware-based fault attack against the RSA verification process was presented in 2005 [155] and generalized one year later [122]. The attack consists of forcing the attacked cryptographic device to use a slightly modified modulus instead of the original one by inducing transient hardware faults. Since the factorization of the altered modulus is known, the attacker can calculate a new private key so that the device accepts the signature of any arbitrary message signed with this new key. A similar attack was presented against the signature process in 2006 [41]. The authors even show how the full RSA private key can be recovered by corrupting the modulus. Hence, it was concluded that "RSA public key elements also have to be protected against fault attacks". In the same year, it was even questioned if it is wise to publish public key elements [82]. The authors present an attack against the RSA verification process where they induce not transient, put permanent faults. This allows the forgery of any signature at any time.

Another work that targets the RSA modulus was published in 2012 [119]. The authors developed a purely software-based fault attack against the verification process. They showed that the modulus can be completely replaced when the structures which manage the public key material are attacked. The new modulus can be easily factorized with a high probability. The practicability was demonstrated on a widely deployed conditional access device.

2.4.2 Elliptic Curve Cryptography

Fault attacks have also been presented against Elliptic Curve Cryptography (ECC). In 2000, the first fault attack on elliptic curve cryptosystems was presented [24]. The authors present three ideas for faults attacks, all of which are based on the same idea: when the coordinates of a point are modified by a fault, the new point will not be on the original curve. The curve on which the modified point lies is potentially cryptographically less secure. Hence, the ECDLP might be easier to solve on that curve. The authors stress that this attack might even be possible without any fault induction: if the DUA does not explicitly check whether or not the input point is on the specified curve, a malicious user can just input a point with the desired

properties. Later, this work was refined to a more relaxed fault model [50]. Another idea for fault attacks against ECC are sign change attacks [35], in which the signs of intermediate points are changed to facilitate the computation of the secret scalar. It is more difficult to mitigate these attacks, since the modified points still lie on the original curve, so that integrity checks would not detect the fault after such attacks. Sign change fault attacks were also described against PBC [180]. In 2008, an attack tailored to the Montgomery ladder was presented [69]. The authors state that they can reveal the secret scalar with only one or two faults, even in the presence of countermeasures which aim at preventing fault attacks. The attacks also consist in performing operations with the modified point on another curve. In this scenario, the modified point lies on the twist of the original curve.

2.4.3 Symmetric Cryptography

Fault attacks were also conducted against symmetric cryptography. Still in 1997, Biham and Shamir described the idea of these kinds of attacks against symmetric cryptographic algorithms [27]. They applied them against the DES and Triple-DES ciphers. Such an attack is called Differential Fault Analysis (DFA), or Differential Fault Attack [136]. The attacker induces faults on the cryptographic primitive level. The assumed fault model gives her partial information about the differences between certain states of the correct and the faulty computations, although she will not know the concrete value of the fault in most scenarios. Since the attacker also knows the correct and faulty ciphertext, and thereby their difference, she can deduce information about the secret key. Small differences in the fault models might crucially affect the capabilities and the complexity of the attacks [27]. Today, there is a wide range of literature on DFAs, e.g., [16, 107, 145].

The mathematical security of block ciphers increases with the number of rounds. Hence, another line of research on fault attacks aims at reducing the number of rounds which are actually computed. In [47], the authors reduced the number of rounds of an AES computation to one by attacking several instructions by means of power glitches.

Chapter 3

Photonic Emission Analysis

Low-cost evaluation methods [...] must be considered in security evaluations.

(S. Skorobogatov [160])

Not only S. Skorobogatov called for low-cost evaluation methods. To show that photonic side channel attacks are a real threat to cryptographic devices, other researchers suggested further research in this direction as well: "An interesting point for future research will be to re-do these experiments on [...] low-cost systems to validate the real benefit of them" [59]. We accomplished this task and showed that successful photonic side channel attacks do not require an awfully expensive measurement system.

The following description is based on [104, 105, 149, 150]. We explain the physical aspects of the photonic side channel in this chapter and continue with the explanation of the cryptographic aspects in Chapter 4. The development of the setups for the measurement of photonic emissions was done by E. Dietz, S. Frohmann, D. Nedospasov, and A. Schlösser.

This chapter is divided into two sections. In Section 3.1, we first explain the physical background of photonic emission and detection techniques for their measurement. We review applications of photonic emissions for failure analysis, cryptography, and reverse engineering. In Section 3.2, we explain in detail the low-cost setups that we used for our photonic side channel attacks. We start with the description of our target devices and then explain the components of the setups and the methodology for emission images and for spatial and temporal analysis.

3.1 Photonic Emission

Photonic Emission Analysis (PEA) exploits the fact that photonic emissions of a device built in CMOS technology depend on the data it processes and the operations it performs. Contrary to other physical characteristics which depend on the data and the operation, like instantaneous power consumption, however, PEA does not only provide high temporal resolution, but also spatial orientation and high spatial resolution. The spatial resolution of PEA can go down to transistor level. This allows for more sophisticated and less preventable attacks.

3.1.1 Photonic Emission in CMOS

Photonic emission in silicon has been investigated since the 1950s [49, 125]. First observed in p-n junctions under reverse bias conditions, it was later discovered that Metal-Oxide-Semiconductor Field-Effect Transistors (MOSFETs) exhibit optical emission that is caused by similar effects [108, 167, 168]. Since then, this has been extensively used in IC failure analysis because of the close correlation between the electronic condition of the device and the generated photons.

Photons are generated when a transistor switches and current flows. In CMOS technology, photoemission can be observed from a transistor's channel due to a process called hot-carrier luminescence. Carriers travelling the conductive channel are accelerated by the electric field between source and drain. At the drain edge of the channel, in the pinch-off region, this energy can be released via phonon-assisted radiative transitions, generating photons [178]. The spectral distribution of the photons follows the energy distribution of the generating electrons, which is spread widely by different effects in the channel. Thus, photon wavelengths reach from the visible to the infrared (IR) with a peak at near-infrared (NIR) wavelengths just above 1 µm. Hot-carrier luminescence is a very weak effect: only a tiny amount of electrons contribute to the generation of photons. The photon generation rate is overproportional to the supply voltage and proportional to the transistor switching frequency. However, the actual emission probability is complex to calculate as many factors contribute [134]. Scaling, for example, has some positive effects on the emission probability [169], which makes it possible to observe hot-carrier luminescence even in recent technologies.

Hot-carrier luminescence is dominant in n-type transistors due to the higher mobility of electrons as compared to holes. In p-type transistors, the holes are the majority charge carriers and consequently, the emerging hot-carrier luminescence is weaker. Hence, 0-1 transitions might be better detectable than 1-0 transitions. Figure 3.1 illustrates the photonic emission from a switching CMOS inverter with n-type and p-type transistors. Light emission is denoted with $h\nu$. Since the silicon substrate is n-type, the n-type transistor is manufactured in a p-well.

Optical emissions of CMOS logic show a data-dependent behavior similar to power consumption. However, in contrast to power consumption, photonic emission is a statistical process and measurements result in discrete count numbers. In addition, the absolute number of detectable photons is very low and needs to be integrated over many switching operations. Approximately every 10^5th to 10^6th switching operation led to the emission and detection of a photon during our experiments.

3.1.2 Detection of Photonic Emission

Since multiple interconnect layers of modern IC designs prevent generated photons from escaping the IC on the frontside, hot-carrier luminescence is best observed from the backside. In this case emitted photons have to pass through the silicon substrate.

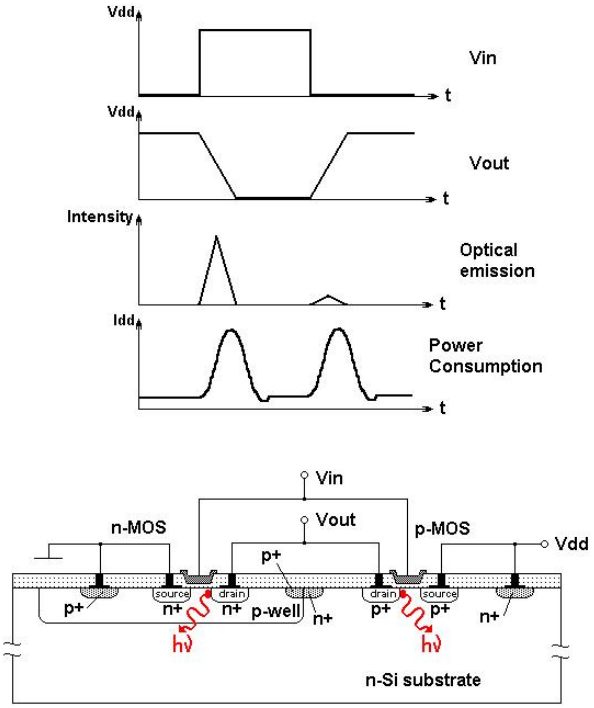


Figure 3.1: Photonic emission from a switching CMOS inverter with n-type and p-type transistors (by courtesy of S. Skorobogatov [160]).

Silicon, however, is highly absorptive for wavelengths shorter than the bandgap energy, which is approximately 1050 nm. Thus, depending on the thickness of the substrate, only a small number of the visible spectrum (VIS) and NIR photons reach the surface and can be captured for analysis. To reduce absorption, the substrate can be mechanically thinned. This can be done with standard backside polishing machines and even with much simpler polishing techniques [126].

Detection Techniques

Semiconductor-based detectors are the technology of choice to measure photoemission. These detectors are based on the quantum photoelectric effect: a photon excites an electron-hole pair which contributes to the photocurrent [164].

Detection of hot-carrier luminescence from the backside must overcome two issues. First, high spatial and temporal resolution is not provided at the same time by a single detector. Charge-coupled devices offer high spatial resolution with a high detection sensitivity, i.e., a high quantum efficiency. In a Charge-Coupled Device (CCD), quantum efficiency is defined as the number of charge carriers produced per photon, i.e., it is a measurement of a device's electrical sensitivity to light [164]. However, CCDs rely on integration of incoming signals over time. Hence, they allow

only slow frame rates and time-resolved measurements are not possible. Single-pixel detectors like Photo Multiplier Tubes (PMTs), Avalanche Photo Diodes (APDs), and Superconducting Single Photon Detectors (SSPDs), in contrast, offer picosecond timing resolution - but only for one small detection area. Hence, both CCDs and single-pixel detectors provide either high temporal or high spatial resolution, but none of them offers both at the same time. Second, readily available and affordable Si-based detectors cover only a fraction of the relevant NIR spectral range. Thus, for efficient photonic emission analysis more complex and expensive solutions, such as InGaAs-based detectors, are necessary. This is especially true for analysis of modern ICs with small feature sizes, as the emission spectrum shifts further to the infrared with decreasing transistor gate length. This technology, however, is more expensive and carries the disadvantage of noisy behavior.

The methodology of choice to perform spatially and temporally resolved measurements via backside analysis is Picosecond Imaging Circuit Analysis (PICA). Integrated PICA systems are based on gated multi-channel plates with NIR-sensitive cathode materials. They were developed explicitly for failure analysis of semiconductors [98, 172]. PICA delivers both spatial and very high temporal resolution, but offers only very limited NIR-sensitivity [98, 172]. However, integrated PICA systems have a starting price of around 1,000,000 €. Since it is unlikely that these systems will ever become a commodity, this price will most probably not decrease within the next years.

3.1.3 Applications of Photonic Emission

Since around 25 years, different technical and scientific areas benefit from the analysis of photonic emission [160]. We present related work for the application of photonic emission for failure analysis, cryptography, and reverse engineering.

Photonic Emission for Failure Analysis

In the failure analysis community, hot-carrier luminescence has primarily been used to characterize implementation and manufacturing faults and defects [62, 156]. Here, the technologies of choice to perform backside analysis are PICA [21] and SSPDs [161]. Both technologies are able to capture photonic emission with high performance in their respective field, but carry the downside of immense cost and complexity.

Photonic Emission for Attacking Cryptography

Here, we only explain the physical and measurement-related aspects of photonic emission for cryptographic applications. The mathematical and cryptographical aspects are discussed in Chapter 4.

One of the first uses of photonic emissions in CMOS in a cryptographic application was presented in 2008 [67]. However, the authors increased the power supply to 7 V operating voltage, which is above the chip's maximum limit for voltage. The authors utilize PICA to spatially recover information about binary additions ($\oplus$)

related to the AddRoundKey operation of AES running on a 0.8 µm PIC16F84A microcontroller. As the authors state, such a PICA device "is available in several laboratories, for example, in the French space agency CNES". Employing PICA in this manner led to enormous acquisition times. This is especially true considering the size of the executed code. It took the authors 12 hours to recover a single potential key byte [67]. In the same time our first system recovered all 16 bytes of the 128-bit AES key twice, see Section 4.1.1, although our first measurements were not time-optimized, but only for the Proof of Concept (PoC). Later, we developed an improved system which needs only some minutes to capture all emissions necessary for a successful 16-byte key recovery.

In 2011, an integrated PICA system and laser stimulation techniques were used to attack a DES implementation on an FPGA [59]. The authors show that the optical side channel can be used for differential analysis. They partly recover the secret key using temporally resolved measurements. As the authors note, the use of equipment valued at more than 2,000,000 € does not make such analysis particularly relevant.

Photonic Emission for Reverse Engineering

In 2009, photonic emissions were analyzed for basic reverse engineering [160]. Low-cost equipment was used to capture photonic emissions via backside analysis and gain basic information about the operations executed on an IC. The author compares the information leakage gained from the power supply line and from optical emission. Even though the author presents low-cost solutions for both spatially and temporally resolved photonic emission analysis, no attacks using temporal information are demonstrated.

In 2012, an automated methodology for performing functional analysis of integrated circuits was introduced [123]. By selectively executing code on a given chip, the resulting optical emission images yield critical information about the chip's functional layout. This methodology provides an efficient way to isolate potential points of interest and can also serve as a basis for photonic side channel attacks, as we will describe in Chapter 4.

3.2 Experimental Setups

To emphasize the relevance of photonic side channel attacks, a low-cost opto-electronic setup to perform the necessary measurements was developed. To increase the relevance of optical vulnerability analyses and attacks, the experimental system was constructed with off-the shelf components and employs readily available technical solutions. As neither Si- nor InGaAs-based detectors can deliver both spatial and temporal resolution in the NIR range at the same time for an affordable price, our system combines the inherent advantages of both detector technologies in an integrated system. The overall complexity and cost of this system is considerably lower

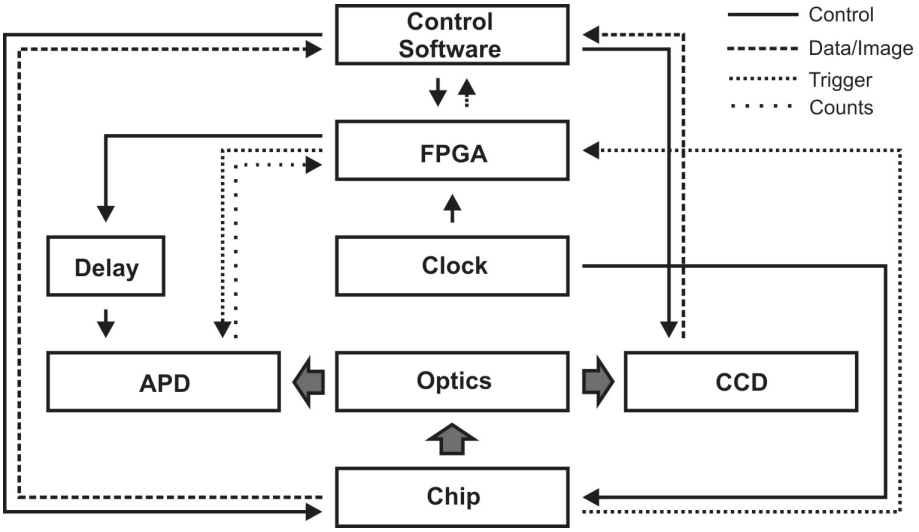


Figure 3.2: The NIR microscope connects the chip under observation to the two detectors (APD and CCD). These are controlled via an FPGA-based controller which handles gate synchronization and delay control as well as time-to-amplitude conversion and multichannel counting.

than common semiconductor failure analysis systems. The price for the necessary hardware adds up to approximately 50,000 - 60,000 €.

The experimental setup consists of two detectors optically and electrically connected to the DUA via a custom-built microscope and an FPGA-based controller, see Figures 3.2 and 3.3. The microscope is optimized for wavelengths between 800 and 1600 nm, i.e., the NIR range. The DUA is soldered onto a custom Printed Circuit Board (PCB) and mounted on travel stages. The microscope itself uses finite conjugate reflection type objectives with high numerical aperture and gold plated mirrors to achieve maximum throughput and a spectrally flat transmission curve. Photons emitted from the DUA are collected by the microscope objective and separated by a dichroic beamsplitter. Photons with wavelengths shorter than 1 µm pass through the beamsplitter and are imaged onto a Si-CCD, whereas photons with longer wavelengths are reflected onto an InGaAs-APD. The first detector, the Si-CCD, is described in detail in Section 3.2.2, and the second detector, the InGaAs/InP-APD, is described in detail in Section 3.2.3.

3.2.1 The Target Devices

We conducted attacks against two different target devices, an ATMega328P and an ATXMega128A1. Both DUAs ran an assembler implementation of the first two AES operations, the initial `AddRoundKey` operation and the `SubBytes` operation of the first round. After these two operations, the input was reset and the measurement restarted. Both stack- and heap-based AES implementations were tested,

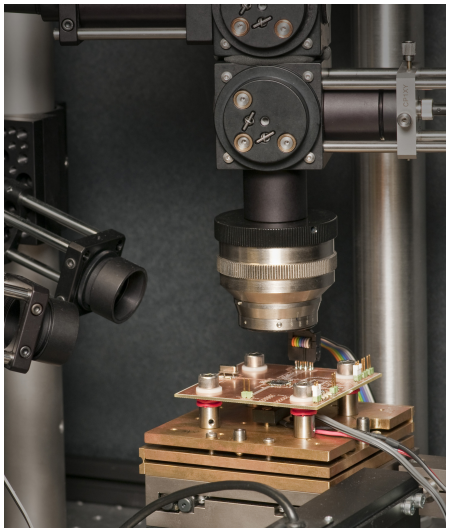


Figure 3.3: In our opto-electronic setup, the chip under observation is mounted upside down on a custom PCB underneath the microscope objective.

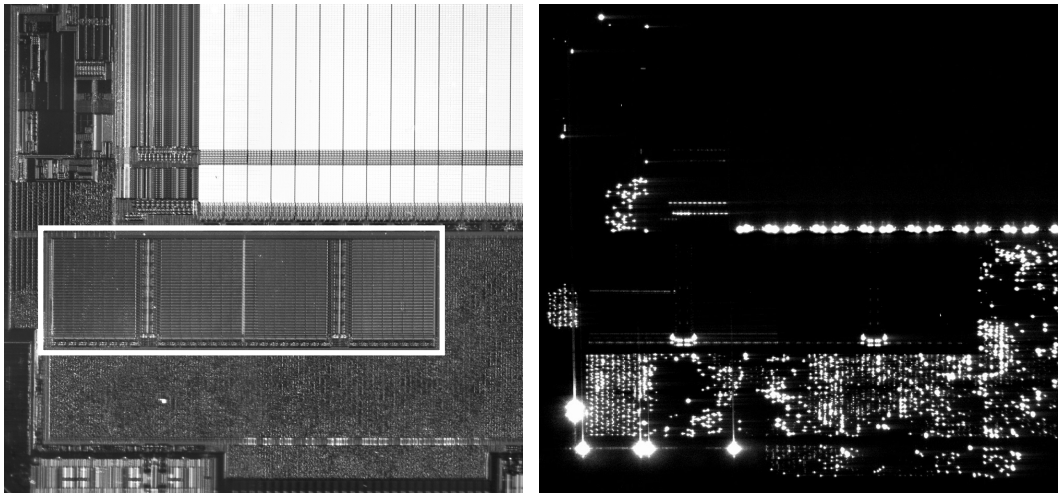
but did not lead to differences in relevant parameters. The AES S-Box was implemented in the microcontrollers' data memory, i.e., the Static Random-Access Memory (SRAM). The two DUAs differ in the size and layout of their SRAM, which is structured in rows and columns.

ATMega328P

The first DUA was an ATMega328P microcontroller [1]. It has four 512-byte SRAM banks, see Figure 3.4(a), each of which is made up of 64 rows and 64 columns. Thus, each row stores a total of 64 bits, or 8 bytes. The ATMega328P was inversely soldered to an evaluation board and supplied with 16 MHz quartz oscillator and 5 V operating voltage, as well as decoupling capacitors and an I/O header for external communications and programming. To shorten the emission image acquisition times, the backside of the ATMega328P was mechanically polished with an automated backside sample preparation machine. The remaining substrate was approximately 25 µm thick.

ATXMega128A1

The second device was an ATXMega128A1 [2]. The ATXMega128A1 has eight 1024-byte SRAM banks, see Figure 3.5(a), and thus a total of 8 kilobytes of SRAM. Each bank is made up of 64 rows and 128 columns, corresponding to 16 bytes per row. Though this IC includes a hardware AES implementation, it executed the same software AES routine as the ATMega328P. The ATXMega128A1 microcontroller was again inversely soldered to an evaluation board but supplied with just 3.3 V in all our experiments. The clock source for our experiments was the internal 32 MHz



(a) Reflected light image of ATMega328P SRAM. (b) Emission image of ATMega328P SRAM.

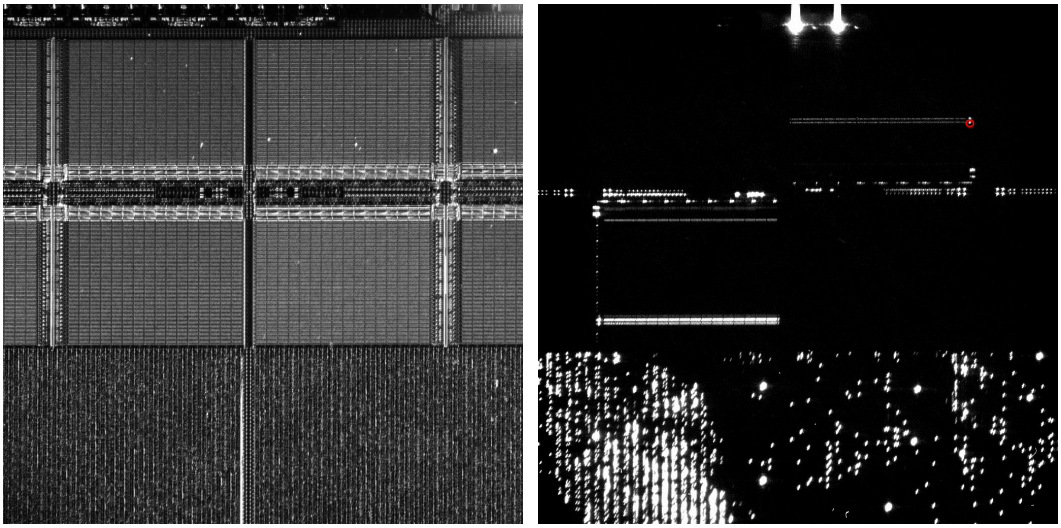
Figure 3.4: Reflected light and 120 s emission images of the ATMega328P SRAM with 6.3-fold magnification. The four SRAM banks are marked with a white rectangle in Figure 3.4(a). The PoC AES implementation is running on the chip.

RC oscillator. As with the ATMega328P, the backside was mechanically polished to approximately 25 µm.

3.2.2 Emission Images

A single Si-CCD serves as the primary detector and captures VIS and NIR photons below the silicon bandgap energy of approximately 1050 nm. The camera uses a one megapixel deep depletion sensor, which is back-illuminated and thermoelectrically cooled to about $-70\,^{\circ}$ to ensure optimal NIR sensitivity and low dark current rates. This detector can take darkfield reflected light as well as emission images through the substrate silicon. Together with a high numerical aperture objective, the system offers diffraction-limited spatial resolution below 1 µm. The acquisition time necessary for adequate emission images ranges from a few seconds to many minutes. It depends strongly on the supply voltage and the structure size of the DUA, the switching frequency of the transistors under observation, and the substrate thickness. Optimized software implementations on the DUA can increase the execution loop frequency, which often reduces acquisition times to seconds [123].

Figure 3.4 shows a reflected light image and an emission image of the ATMega328P, and Figure 3.5 shows the respective images for the ATXMega128A1. By studying emission images of the DUAs and with the techniques introduced in [123], relevant components of a microcontroller can be identified. Since both the SPEA as well as the DPEA from Chapter 4 target the SubBytes operation with S-Box table lookups, we concentrated on finding the S-Box.



(a) Reflected light image of ATXMega SRAM. (b) Emission image of ATXMega SRAM.

Figure 3.5: Reflected light and 300 s emission images of the ATXMega128A1 SRAM with 10-fold magnification. The eight SRAM banks are visible in the top part of Figure 3.5(a). Figure 3.5(b) shows two highlighted lines in the middle right bank of the upper row. These correspond to accesses to the first two elements of the AES S-Box. The row driver whose emissions are measured for the SPEA is marked with a red circle.

The S-Box can easily be identified in memory and is shown in detail in Figure 3.6. In the SPEA, we exploit emissions of the row drivers. Their emissions are clearly visible to the left of the memory bank. The figure also reveals that emission images allow direct readout of the stored bit values. Figure 3.6 shows that the bytes are not stored one after the other. The storage layout keeps the lsb of all bytes from a certain row at the left, and the msb at the right. Thus, to read a certain byte in such a memory with 8 bytes per row, one has to read from left to right every eighth bit, see Figure 3.6. Thus, emission images allow direct readout of the stored SRAM data. In general, this is a threat for secure devices. In case of the AES S-Box it is no threat since the stored S-Box elements, i.e., the S-Box outputs, are publicly known anyhow. For the SPEA presented in Section 4.1, however, it is important to know exactly how the S-Box is stored, i.e., how many bytes span a row, and which is the address of the first S-Box element. The attacker needs to know whether or not a certain value is stored in a selected row.

3.2.3 Spatial and Temporal Analysis

For a photonic side channel attack, we want to observe only a small part of the DUA and get its photonic emissions with a high temporal resolution. We identify this part

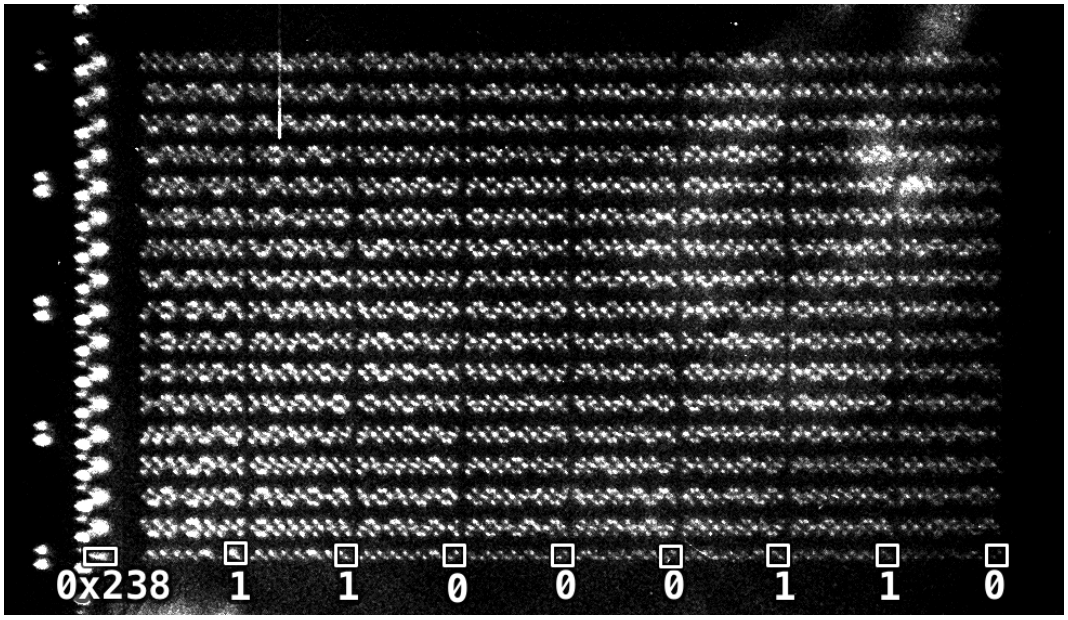


Figure 3.6: Optical emission image of the S-Box in memory. The 256 bytes of the S-Box are located from memory address 0x23f to 0x33e, as in Table 4.1. The address 0x23f is the eighth byte of the SRAM line starting with address 0x238, i.e., the S-Box has an offset of 7 bytes. The emissions of the row drivers are clearly visible to the left of the memory bank. The image allows direct readout of the bit values of the stored data. The byte shown in the overlay, for example, corresponds to 0b01100011 = 0x63, the first value of the AES S-Box.

from emission images captured with the CCD and then use the second detector to measure the photonic emission only of the selected components. During our research, we used two different techniques for the temporally resolved measurements.

Both are based on a single InGaAs/InP APD commonly found in telecommunication transceivers (Telcordia GR-468-CORE). It is operated in Geiger mode, i.e., the diode is supplied with a high reverse-bias voltage just below avalanche breakdown. Single photons exiting one electron within the diode then cause an avalanche of carriers and thus high amplification. To minimize dark current, the diode is thermoelectrically cooled. Dark current (or dark counts) is the leakage current when the diode detects photons although it is not exposed to the light source [164]. Afterpulsing, an adversary effect in InGaAs/InP APDs, is reduced by extensive quenching circuits and gated operation, a technique where the bias voltage is only applied during small windows in time. The diode is coupled to the microscope via an optical fiber. The measuring spot of the DUA, corresponding to the fiber's aperture, can be freely positioned for temporal analysis. The spot size can be varied. Areas of interest, identified in an emission image, can thus be selected for temporal analysis with high spatial selectivity. Even single transistors can be selected for precise measure-

ments. Because of its spectral sensitivity above 1 µm, unlike the CCD, this detector does not require a thinned DUA substrate, as silicon is transparent in this spectral range. Hence, if spatial orientation relative to the IC's layout can be obtained by other means, substrate thinning can be omitted completely. This is interesting when applying the presented methodologies across multiple samples of an identical IC, as only a single sample has to be prepared to provide orientation.

Once the area of interest is identified, we record a set of traces while the device encrypts certain plaintexts. In the case of a DPA, a trace refers to a set of power consumption measurements taken across a cryptographic operation [103]. Such a trace can be plotted as a line graph. In the case of photonic emission analysis, a single measurement only results in a discrete vector with most of the entries equal to 0. However, we also use the term trace to refer to the measurement of photonic emissions of a single encryption of a certain plaintext. Hence, a trace $\mathbf{t}_i$ is recorded while the device encrypts the input data i. Each trace consists of N points in time, i.e., N is the length of the traces and thus, $\mathbf{t}_i = (t_{i,1}, \ldots, t_{i,N})$. The traces $\mathbf{t}_i$ and their components $t_{i,n}, n \in \mathcal{I} = \{1, \ldots, N\}$, respectively, thus refer to real photonic emissions and each $t_{i,n}$, corresponds to a number of count events.

At the beginning of our research, we had to compose each trace from multiple measurements. When it is used in so-called gated operation, the APD is rendered sensitive only for a short window in time during every execution of the target code. This short time frame is called detection gate. To reconstruct the complete signal temporally, the detection gate has to be synchronized during all executions and it has to be shifted regularly relative to the signal, similar to a sampling oscilloscope. Provided that the detection gate can be controlled with high precision, the time resolution and the measurement time depend only on the employed gate width.

To implement this detection scheme, we used an FPGA-based controller that is phase-locked to the clock of the DUA. A block diagram of the system is shown in Figure 3.2. When the DUA executes the target code, the FPGA digitally delays and triggers the APD detection gate. Each FPGA-controlled trigger renders the APD sensitive for a preset amount of time, the detection gate. The detection gate length was 20 ns for the SPEA measurements. Detection events are sent back to the FPGA and counted in the corresponding time bins. The set of triggers is shifted by the length of the detection gate after an appropriate number of measurements and the next set of data points is being collected. An additional analog delay can be employed for fine delay control. The absolute time resolution of the system is jitter-limited to approximately 1 ns. The measurement time to reconstruct the extremely weak photoemission signals can be immense: millions of measurements may be necessary to achieve an adequate Signal-to-Noise Ratio (SNR), i.e., a large signal and low noise. To drastically reduce the measurement times, the FPGA triggers hundreds of APD detection gates per execution of the target code. This results in interleaved measurements, and the whole set of interleaved measurements then reconstructs the signal. With our first setup, we stored only accumulated measurements, i.e., we stored only a single trace per input byte. This was more efficient in terms of memory space, but obstructed efficiency analyses from the cryptographic point of view.

Improved Setup

Later, we developed an improved system for the temporal measurement of location-dependent leakage. This was the consistent further development of the original setup. It was also used in [166]. This system does not require multiple detection gates, but allows to measure the traces in free-running mode. Moreover, it allows us to store each trace separately, instead of one accumulated trace per input. This allows more detailed analyses.

Due to the weak photonic emission from the DUA, temporal acquisition of this leakage requires a detector which is very fast and highly sensitive to infrared wavelengths. The improved setup features better adopted optics and an InGaAs-APD that is more suitable for this kind of application, as compared to the original setup. The detector signal from the APD is processed first in a Time-to-Digital Converter (TDC), before transmitted to the data processor together with the trigger signal from the DUA. The TDC tags each occurring event with a resolution of 81 ps. This allows for the calculation of the timing of a detected photon in relation to the DUA's trigger. The accuracy of the measured timing of single events is limited by the overall jitter of about 200 ps, which is caused by the electronics in the APD and the signal detection at the TDC. If required, superresolution methods can even increase the time resolution to 6 ps. However, regarding the clock period of 62.5 ns of the DUA, a time resolution of about 200 ps is sufficient. The low dark count rate of the APD allows to use it in free-running mode. Therefore, the costly gating technique as used in the original setup is not longer necessary. Moreover, the free-running mode and the low dark count rate enable a significant reduction in measurement time. We needed only a few minutes to capture all traces for a successful DPEA, see Section 4.2. Nevertheless, the improved system has also been built from commercially available components. Only the mounting of the DUA on a three-dimensional moving stage and the necessary electronics to communicate with the DUA are custom-made. The price for the necessary hardware of our improved system adds up to approximately 50,000 - 60,000 €. Thus, its price is similar to the price of the original setup.

Chapter 4

The Photonic Side Channel

Optical emission attacks will very likely result in the need to introduce new countermeasures during the design of semiconductor chips.

(S. Skorobogatov [160])

This chapter describes how the physical information gained during a photonic emission analysis can be used to reveal information about secret cryptographic keys and to conduct a photonic side channel attack. As it is the case for the well-known Simple Power Analysis and Differential Power Analysis, both simple and differential attacks based on photonic emission are defined and presented. Accordingly, these attacks are called Simple Photonic Emission Analysis and Differential Photonic Emission Analysis.

Definition 4.1 (Simple Photonic Emission Analysis)**.** *A Simple Photonic Emission Analysis (SPEA) reveals secret information of a cryptographic device based on traces of photonic emissions that have been recorded while the device encrypts, decrypts, or signs input data. A single input might be sufficient for a successful SPEA. The information is revealed by mapping certain computational operations to location-dependent leakage and photonic emission patterns at certain points in time, which do not have to be known in advance.*

Note that while it might be sufficient to gain information about photonic emission of only a single input, this does not imply that a single trace is sufficient for an SPEA. As explained in Chapter 3, a single trace does not contain more information than just few photons, if any. Thus, for any analysis, measurements have to be repeated thousands of times.

The first SPEA-like attack was conducted in 2008 [67]. The authors spatially recovered information about an 8-bit XOR operation ($\oplus$). They used the four different constants `0xff` (`0b11111111`), `0xaa` (`0b10101010`), `0x55` (`0b01010101`), and `0x00` (`0b00000000`) as key byte representatives. The values of these constants were revealed when they were xored to state bytes. The values of the state bytes did not have to be known. Both bit transitions 0-1 and 1-0 were detectable in the emission images. Thus, a flipping bit was observable and led to a 1 as key bit, while a non-flipping bit did not lead to photonic emissions and thus was interpreted as a 0. Note that here, whole emission images were analyzed.

In 2009, photonic emissions were used to gain basic information about the operations executed on an IC [160]. Again, no attacks using temporal information were demonstrated, but emission images for $\oplus$ operations were successfully analyzed.

Definition 4.2 (Differential Photonic Emission Analysis)**.** *A Differential Photonic Emission Analysis (DPEA) reveals secret information of a cryptographic device based on traces of photonic emissions that have been recorded while the device encrypts, decrypts, or signs different input data. The data dependency of the intensity of the photonic emissions at certain points in time, which do not have to be known in advance, is exploited by a statistical analysis.*

In 2011, a fragment of the DES algorithm on an FPGA was attacked with a differential photonic emission analysis [59]. The authors partially recovered the secret key using temporally resolved measurements. However, the analysis strongly relied on a specific implementation of DES in which inputs were zeroed. Full recovery of a 64-bit DES key was not presented.

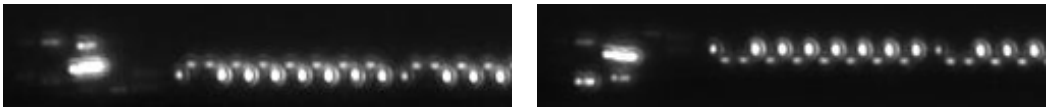
Prior to the research presented in this work, no photonic side channel attack successfully revealed a complete key. All presented attacks till then can be seen as PoC attacks which only expose the general principle of this side channel. They did not reveal full keys, but only demonstrated the method for certain bytes.

We will now present our results of photonic side channel attacks [104, 105, 149, 150]. We attacked AES-128 and achieved full AES key recovery for SPEA as well as for DPEA. For both practical attacks, we used 256 pairwise different 128-bit input messages m_i, $i \in \{0, \ldots, 255\}$ to reveal the secret key. Each of these messages consists of 16 equal bytes, i.e., we have $m_i^j = m_i^{\tilde{j}}$ for all $j, \tilde{j} \in \{0, \ldots, 15\}$. We chose the input messages so that $m_i^j = i$ for all $i \in \{0, \ldots, 255\}$ and $j \in \{0, \ldots, 15\}$.

We start with the detailed explanation of the Simple Photonic Emission Analysis in Section 4.1. This part is based on [149] and [150]. We present the results of the practical attack against AES-128 and show how the attack can be extended for AES-192 and AES-256. We also explain how this attack can be conducted against other cryptographic algorithms. Countermeasures to prevent such attacks are discussed. In Section 4.2, we go on with a detailed explanation of the Differential Photonic Emission Analysis. This part is related to [104] and [105]. We explain how a secret AES key can be revealed with different statistical distinguishers and how such an attack can be prevented.

4.1 Simple Photonic Emission Analysis

The following section explains Simple Photonic Emission Analysis in detail. It is based on [149] and [150]. First, we touch the relevant parameters of the physical attack. Next, we explain the cryptanalytic details of SPEA. We give examples for the attack against AES-128, AES-192, and AES-256 and sketch how the attack can be transferred to other cryptographic algorithms. Finally, we show why certain cryp-



(a) Access to address 0x300. (b) Access to address 0x308.

Figure 4.1: Emission images of memory accesses to two adjacent SRAM rows obtained with the Si-CCD detector. The images were integrated over 120 s.

tographic countermeasures are not effective in case of SPEA and outline potential effective countermeasures.

4.1.1 Physical Attack

Address and access logic is implemented similarly across all platforms and memories. Hence, it is a particularly interesting and relevant target. In our implementation, the S-Box is contained within the SRAM, which led us to consider possible side channels that exist within this memory and for the SubBytes operation. We considered how SRAM is accessed in general and how the S-Box would be accessed in the PoC AES implementation. By observing time-resolved access patterns to a specific row within the S-Box, the set of key candidates can be greatly reduced. This will be explained in the next Section 4.1.2. An analogue analysis can also be applied to the column addresses.

Memory is organized with rows and columns. An SRAM cell read begins with assertion of the word line [139]. When any cell within a row of memory is accessed, the word line for the entire row is asserted. This row-select signal is driven by an inverter that is part of the final stage of the row decode logic [179]. We used the Si-CCD detector and techniques introduced in [123] to analyze the emissions of these memory accesses for different addresses and values. By studying the emission images of our PoC AES implementation, we identified the S-Box within memory. The emission image of the S-Box is shown in Figure 3.6. The emissions of the row drivers are clearly visible to the left of the individual memory lines. With spatial resolution, accesses between even two adjacent rows of SRAM can be clearly differentiated, see Figure 4.1. The assembly code of the SubBytes operation of this implementation is given in Table 4.8.

The emission image of the S-Box in the ATMega328P, see Figure 3.6, reveals that the S-Box spans 33 rows of the SRAM and not $32 = 256/8$ as expected. For the ATXMega128A1 with a memory line width of 16 bytes, the S-Box spanned 17 lines of memory. Since the S-Box was implemented as an array of bytes within data memory, its start address depends on how many other variables are allocated within this memory. It is therefore only with probability 1/8 and 1/16, respectively, aligned to the beginning of a memory row. During our experiments, the S-Box always exhibited an *offset* unless the address was set explicitly.

Provided that the memory line width w is a power of 2, i.e., $w = 2^n$ for some

$n \in \{0, \ldots, 8\}$, the memory line width w and the offset o between the beginning of a memory line and the first S-Box element, both counted in bytes, define the number L of memory lines taken up by S-Box elements:

$$L = \begin{cases} \frac{256}{w} & \text{if } o = 0 \\ \frac{256}{w} + 1 & \text{if } o \in \{1, \ldots, w-1\} \end{cases}$$

Contrary to most textbook representations, e.g., [131], the S-Box elements are not stored top-down as in Table 2.1, but the other way round, i.e., bottom-up. Thus, the S-Box element which gets accessed for input `0x00` is stored in the leftmost position in the bottommost row 1, while the S-Box element which gets accessed for input `0xff` is stored in the rightmost position of the topmost row L. During our attacks and for our fragmentary AES implementation, the S-Box started at address `0x23f` in case of the ATMega328P, and it had an offset of $o = 7$. Table 4.1 shows the corresponding S-Box memory layout. The ATXMega128A1 also exhibited a maximal offset, i.e., $o = 15$. Its start address was `0x215f`.

The accesses to the first two rows of the S-Box on the ATXMega128A1, corresponding to state values `0x00` and `0x01`, are clearly visible in the emission image Figure 3.5(b). Such a row access results in a clearly defined photon detection peak when we use the APD to capture the emission of a single row driver, see Figure 4.2. The same result is obtained for the ATMega328P, see Figure 4.3. Both figures show the time-resolved photonic emission traces, captured during the first `SubBytes` operation at the row driver of the bottommost row, which stored only a single element.

Expecting input-dependent accesses to the memory rows, we set our APD detector to measure the photonic emissions from the row driver inverter corresponding to a fixed SRAM row with S-Box elements. Emission traces for 256 chosen input messages were captured. Note that with both DUAs, we did not completely encrypt the messages, but calculated only the first two operations `AddRoundKey` and `SubBytes` with our fragmentary AES implementation. The measurement of the emissions of a row driver on the ATMega328P required an acquisition time of 90 s for each input message. Thus, the time to fully capture the photonic emission signal over time and all possible inputs, as seen in Figure 4.3, amounted to just over six hours. For a measurement of an ATXMega128A1 row driver a higher number of detection events was necessary to achieve an acceptable SNR. This is due to the lower supply voltage and smaller feature size of the ATXMega128A1 as compared to the ATMega328P. A single trace took just over two hours. If the DUA has two or more key bytes that are equal, fewer traces might be necessary to reconstruct the key, as seen in Figure 4.2.

Note that, as explained in Chapter 3, a single trace is composed of several measurements due to the detection technology, since for the SPEA, we only used the APD which requires detection gates. Thus, the fragmentary encryption has to be computed several times for each trace. Moreover, when we conducted the SPEA, we only wanted to prove that the photonic side channel is a real threat. Thus, we performed the measurements for all inputs overnight and just repeated all computations very often. Then, only one accumulated trace per input message m_i was stored

	0	1	2	3	4	5	6	7
0x338	f9 (99)	fa (2d)	fb (0f)	fc (b0)	fd (54)	fe (bb)	ff (16)	
0x330	f1 (a1)	f2 (89)	f3 (0d)	f4 (bf)	f5 (e6)	f6 (42)	f7 (68)	f8 (41)
0x328	e9 (1e)	ea (87)	eb (e9)	ec (ce)	ed (55)	ee (28)	ef (df)	f0 (8c)
0x320	e1 (f8)	e2 (98)	e3 (11)	e4 (69)	e5 (d9)	e6 (8e)	e7 (94)	e8 (9b)
0x318	d9 (35)	da (57)	db (b9)	dc (86)	dd (c1)	de (1d)	df (9e)	e0 (e1)
0x310	d1 (3e)	d2 (b5)	d3 (66)	d4 (48)	d5 (03)	d6 (f6)	d7 (0e)	d8 (61)
0x308	c9 (dd)	ca (74)	cb (1f)	cc (4b)	cd (bd)	ce (8b)	cf (8a)	d0 (70)
0x300	c1 (78)	c2 (25)	c3 (2e)	c4 (1c)	c5 (a6)	c6 (b4)	c7 (c6)	c8 (e8)
0x2F8	b9 (56)	ba (f4)	bb (ea)	bc (65)	bd (7a)	be (ae)	bf (08)	c0 (ba)
0x2F0	b1 (c8)	b2 (37)	b3 (6d)	b4 (8d)	b5 (d5)	b6 (4e)	b7 (a9)	b8 (6c)
0x2E8	a9 (d3)	aa (ac)	ab (62)	ac (91)	ad (95)	ae (e4)	af (79)	b0 (e7)
0x2E0	a1 (32)	a2 (3a)	a3 (0a)	a4 (49)	a5 (06)	a6 (24)	a7 (5c)	a8 (c2)
0x2D8	99 (ee)	9a (b8)	9b (14)	9c (de)	9d (5e)	9e (0b)	9f (db)	a0 (e0)
0x2D0	91 (81)	92 (4f)	93 (dc)	94 (22)	95 (2a)	96 (90)	97 (88)	98 (46)
0x2C8	89 (a7)	8a (7e)	8b (3d)	8c (64)	8d (5d)	8e (19)	8f (73)	90 (60)
0x2C0	81 (0c)	82 (13)	83 (ec)	84 (5f)	85 (97)	86 (44)	87 (17)	88 (c4)
0x2B8	79 (b6)	7a (da)	7b (21)	7c (10)	7d (ff)	7e (f3)	7f (d2)	80 (cd)
0x2B0	71 (a3)	72 (40)	73 (8f)	74 (92)	75 (9d)	76 (38)	77 (f5)	78 (bc)
0x2A8	69 (f9)	6a (02)	6b (7f)	6c (50)	6d (3c)	6e (9f)	6f (a8)	70 (51)
0x2A0	61 (ef)	62 (aa)	63 (fb)	64 (43)	65 (4d)	66 (33)	67 (85)	68 (45)
0x298	59 (cb)	5a (be)	5b (39)	5c (4a)	5d (4c)	5e (58)	5f (cf)	60 (d0)
0x290	51 (d1)	52 (00)	53 (ed)	54 (20)	55 (fc)	56 (b1)	57 (5b)	58 (6a)
0x288	49 (3b)	4a (d6)	4b (b3)	4c (29)	4d (e3)	4e (2f)	4f (84)	50 (53)
0x280	41 (83)	42 (2c)	43 (1a)	44 (1b)	45 (6e)	46 (5a)	47 (a0)	48 (52)
0x278	39 (12)	3a (80)	3b (e2)	3c (eb)	3d (27)	3e (b2)	3f (75)	40 (09)
0x270	31 (c7)	32 (23)	33 (c3)	34 (18)	35 (96)	36 (05)	37 (9a)	38 (07)
0x268	29 (a5)	2a (e5)	2b (f1)	2c (71)	2d (d8)	2e (31)	2f (15)	30 (04)
0x260	21 (fd)	22 (93)	23 (26)	24 (36)	25 (3f)	26 (f7)	27 (cc)	28 (34)
0x258	19 (d4)	1a (a2)	1b (af)	1c (9c)	1d (a4)	1e (72)	1f (c0)	20 (b7)
0x250	11 (82)	12 (c9)	13 (7d)	14 (fa)	15 (59)	16 (47)	17 (f0)	18 (ad)
0x248	09 (01)	0a (67)	0b (2b)	0c (fe)	0d (d7)	0e (ab)	0f (76)	10 (ca)
0x240	01 (7c)	02 (77)	03 (7b)	04 (f2)	05 (6b)	06 (6f)	07 (c5)	08 (30)
0x238								00 (63)

Table 4.1: AES S-Box with 8 bytes per row and an offset of 7. Each entry denotes the corresponding input value together with the output value in parentheses. All values are in hexadecimal notation. The sum of the input value and 0x23f yields the entry's address as it was implemented in the ATMega328P.

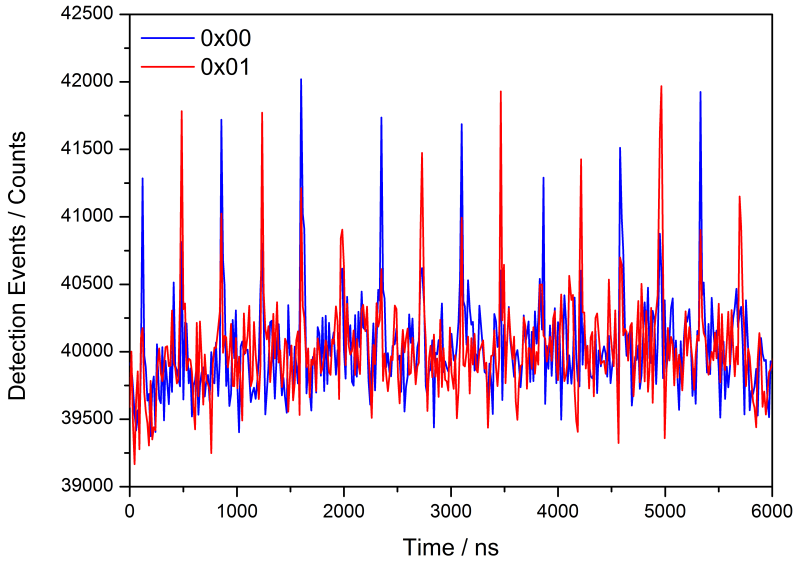


Figure 4.2: Photonic emission traces of the ATXMega128A1. The observed S-Box row stores only the substitution of the byte 0x00. The peaks correspond to the 16 AES state bytes. They are clearly identifiable and show a width of 20 ns, corresponding to the employed detection gate width. The key consists of alternating 0x00 and 0x01 bytes, and so each peak corresponds to 0x00 and 0x01, respectively.

and the usage of the word *trace* in this section therefore contradicts the definition of that word from Section 3.2.3. Hence, with the available set of measurements, we cannot do efficiency investigations concerning the number of necessary traces per input for a successful SPEA.

From our latest measurements with the improved setup, however, we learn that a few thousand measurements per input byte should be sufficient. While our Figures 4.2 and 4.3 show clearly visible peaks with a high SNR due to the long measurement times, these would be less obvious in case one would try to use as few measurements as possible. In this situation, the attacker would empirically define a threshold level which determines whether or not the observed row r of the S-Box was accessed within the measurement. Row accesses can then be identified by determining if the threshold level was exceeded by the photonic emission intensity for the processed byte during a certain period of time. The new setup will need only some minutes to measure all relevant traces for all input plaintexts.

4.1.2 Cryptanalysis

By observing time-resolved access patterns to a specific memory row containing S-Box elements, the set of key candidates can be greatly reduced. A row access

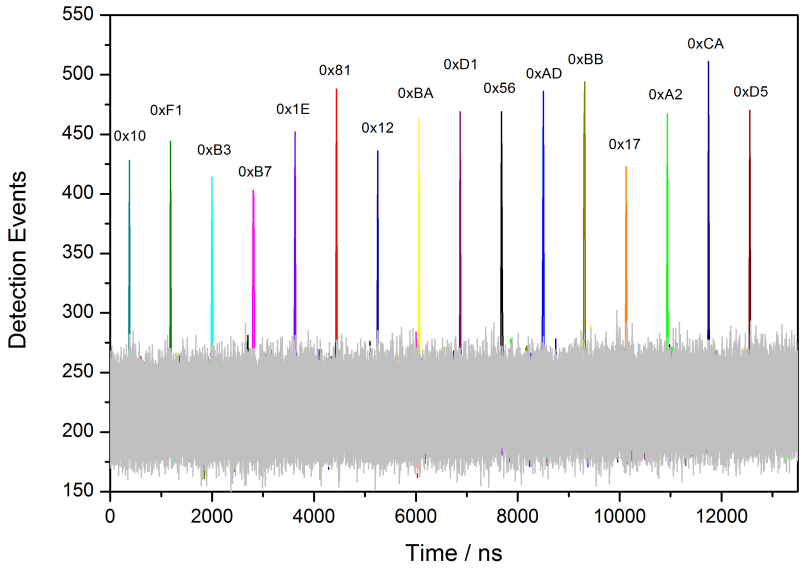


Figure 4.3: Photonic emission traces of the ATMega328P. All possible plaintexts were used when accesses to an S-Box row that stores only the substitution of the byte 0x00 were observed. 16 peaks are clearly identifiable. They correspond to the 16 AES state bytes. Each peak corresponds to a unique key byte, which is annotated.

results in a clearly defined photon detection peak with a high SNR, see Section 4.1.1. These observed row accesses, i.e., the location-dependent leakage, can be used to infer the secret key.

AES-128

We want to reveal the secret key $k \in \{0,1\}^{128}$ with a Simple Photonic Emission Analysis. Therefore, we measure photonic emissions of a selected row driver to learn whether or not a certain row of the AES S-Box was accessed for known input messages. This information helps us to reduce the key space considerably, or even to reveal the secret key directly.

Since AES operates on bytes, the following attack works sequentially for all 16 key bytes k^0 to k^{15}. As explained above, we used 256 input messages m_i, $i \in \{0,\ldots,255\}$ with $m_i^j = i$ for all $i \in \{0,\ldots,255\}$ and $j \in \{0,\ldots,15\}$. Thus, by capturing the emissions of all 16 state bytes in each measurement as explained in Section 4.1.1, all 16 key bytes can be revealed simultaneously. Note that more plaintexts do not increase the attack success if the attacker solely exploits the emissions during the first round. Since here the 16 state bytes are processed independently, there are only $2^8 = 256$ possibilities for each plaintext byte. Thus, we cover all

possibilities for plaintext values with these input messages m_i. On the other hand, fewer plaintexts do not necessarily decrease the attack success.

Recall that this SPEA was the first successful photonic side channel attack which revealed a secret cryptographic key. It was our goal to show the potential threat that this side channel poses to systems with sensitive data. At that point in time, it was not our goal to be as efficient as possible. Hence, we used all possible input plaintexts although this number can be greatly decreased if one uses chosen messages with a certain distribution. Especially, it can be decreased by calculating the set of key candidates on-the-fly. Then, the attacker can stop the attack as soon as she has reached a reasonable set of key candidates.

Without loss of generality, we explain the general analysis only for the first key byte k^0. For a given input value, each observed access can be assigned to a number of key candidates. Hence, for each observed row access, we compute the key values which could have caused the measured access. The attack exploits that the first calculation of the SubBytes is only the second operation of the AES algorithm. We recall that only the initial AddRoundKey operation has been computed before (see the detailed explanation of the AES algorithm in Section 2.2.1). The SubBytes operates on the XOR sum of the input byte and the key byte: $S(m \oplus k)$. Thus, due to the chosen input messages and the bijectivity of the AddRoundKey operation for a fixed key byte, every element of the S-Box is accessed exactly once for every processed byte in the first AES round. However, we do not measure accesses to specific matrix cells, but only to multiple-element rows, i.e., we use the access patterns to a given row to eliminate key candidates.

Let Z denote the set of plaintext bytes which cause a row access, i.e., $\mathrm{Z} \subseteq \{m_i^0 \mid i \in \{0, \ldots, 255\}\}$. Then, for every $m \in \mathrm{Z}$, we compute the set $\widetilde{\mathcal{K}^0}_m$ of key bytes which could have caused the access for that byte. These elements are key candidates corresponding to the processed plaintext. There are exactly as many key candidates as there are S-Box elements in the observed row; those that could have called an address within that memory line.

$$\widetilde{\mathcal{K}^0}_m = \{\tilde{k} \in \mathcal{K}^0 \mid m \oplus \tilde{k} \in \mathrm{R}_r\} \tag{4.1}$$

Here, R_r is the set of state values which cause an access to the selected row r. These elements are the input to the SubBytes operation and thus the result of the initial AddRoundKey operation. Then, we can compute the set $\widetilde{\mathcal{K}^0}$ of candidates for the first key byte k^0 as

$$\widetilde{\mathcal{K}^0} = \bigcap_{m_i \in \mathrm{Z}} \widetilde{\mathcal{K}^0}_{m_i}. \tag{4.2}$$

Since k is used during all accesses for all input values, the correct key byte k will be in the intersection of the sets $\widetilde{\mathcal{K}^0}_m$ of possible key candidates. Regarding the attack success depending on the cardinality of $\widetilde{\mathcal{K}^0}$, we have to differentiate between three cases: the S-Box can either be *aligned*, i.e., it spans $256/w$ rows with

w elements each. If the S-Box is not aligned, it has a positive *offset*. The offset o is the number of memory bytes in the row of the first S-Box element (input `0x00`, output `0x63`) which are not filled with S-Box elements. The offset can be even or odd and is smaller than the number w of bytes in a memory row, i.e., $o \in \{0, \ldots, w-1\}$. In Table 4.1, we show how the S-Box is stored for a memory width $w = 8$ and an offset $o = 7$.

Depending on the offset and the memory width, we can describe the set R_r of elements which cause an access to the selected row r more concretely. For a positive S-Box offset by o bytes, the first row of the S-Box in memory will exhibit $w - o$ accesses, the last row will show o accesses. All other rows again will undergo w accesses, as they do for $o = 0$.

$$\mathrm{R}_r = \begin{cases} \{(r-1)\cdot w, \ldots, r\cdot w - 1\} & & o = 0 \\ \left.\begin{cases} \{0, \ldots, w-1-o\} & r = 1 \\ \{(r-1)\cdot w - o, \ldots, r\cdot w - 1 - o\} & 2 \le r \le 256/w + 1 - 1 \\ \{255 - o + 1, \ldots, 255\} & r = 256/w + 1 \end{cases}\right\} & o \ge 1 \end{cases}$$

We will now discuss the results for all three cases: aligned S-Box, S-Box with even offset and S-Box with odd offset. After this discussion, the numbers of remaining candidates for a 128-bit key for all possible offsets, given a memory line width of 8 bytes, are summarized in Table 4.4. This memory line width was given in the ATMega328P. The respective numbers for an SRAM storing 16 bytes per row can be found in Table 4.5. This memory line width was given in the second device that we analyzed, the ATXMega128A1. Therefore, we assume $w \in \{8, 16\}$ in the following explanation. It is worth noting that, given a fixed offset, the number of remaining key candidates as given in Tables 4.4 and 4.5 cannot be reduced by a second set of measurements at another row.

Aligned S-Box For an aligned S-Box with offset $o = 0$, we can observe $w = 2^n$ accesses for every row with the input bytes outlined above. However, although this allows us to intersect w sets of key candidates with w elements each, we will be left with w candidates per key byte, since all w sets of key candidates are identical. This can be explained by closer looking at the `AddRoundKey` operation and the resulting state bytes which cause an access to a certain row of the S-Box .

We have $\mathrm{R}_r := \{(r-1)\cdot w, \ldots, r\cdot w - 1\} = \{(r-1)\cdot 2^n, \ldots, r\cdot 2^n - 1\}$ for some $r \in \{1, \ldots, 256/w\}$ and $n \in \{3, 4\}$. Independent of the observed row, this implies that all elements in R_r have the same $2^n - n$ msb and differ only in their n lsb. From this it follows that, independent of the observed row and the plaintext byte which leads to a row access, all key candidates have the same $2^n - n$ msb and differ only in their n lsb. Since each plaintext byte leads to exactly $w = 2^n$ key candidates, these candidates comprise all possible bytes with the given $2^n - n$ msb. Thus, identical sets of $w = 2^n$ elements each are intersected and thus, 2^n key candidates remain. The $2^n - n$ msb of each key byte are completely revealed, but an attacker does

not learn anything about the n lsb. From an attacker's point of view, an aligned S-Box is the worst case.

The row which is observed does not influence the success of the attack in case of an aligned S-Box. This can also be seen in Tables 4.4 and 4.5. Since multiple accesses for different input bytes to the same row do not reduce the number of candidates in the case of an aligned S-Box, it is sufficient to observe a single row access per key byte.

Example 4.1. *Let the memory line width $w = 8$ and let the S-Box be aligned. We observe row $r = 12$ and want to reveal the first key byte. Thus, for the given parameters we have $R_{12} := \{88, \ldots, 95\} = \{\texttt{0x58}, \ldots, \texttt{0x5f}\}$. The first access is observed for the plaintext byte* `0x68`*. This leads to eight key candidates: $\widetilde{\mathcal{K}}^0_{0x68} = \{\texttt{0x30}, \texttt{0x31}, \texttt{0x32}, \texttt{0x33}, \texttt{0x34}, \texttt{0x35}, \texttt{0x36}, \texttt{0x37}\}$, since $(\texttt{0x68} \oplus k) \in R_{12}$ for all $k \in \widetilde{\mathcal{K}}^0_{0x68}$. All these candidates have the same five msb* `0b00110`*, but differ in their three lsb. The same eight key candidates can be derived from the next seven row accesses. These are observed for the input bytes* `0x69`*,* `0x6a`*,* `0x6b`*,* `0x6c`*,* `0x6d`*,* `0x6e`*, and* `0x6f`*. Since all eight sets of key candidates are equal, we cannot reveal this key byte uniquely. However, we learned its five msb, which are* `0b00110`*.*

S-Box with Even Offset In the case of an even offset, a maximum of 2^n candidates per key byte remain. Contrary to an aligned S-Box, however, the number of remaining candidates per key byte depends on the observed row and the concrete offset. For an even offset, the number of remaining candidates per key byte can be calculated as follows, depending on n, o, and r:

$$\left|\widetilde{\mathcal{K}}^0\right| = \begin{cases} \min(|w/2 - o|, w/2 - |w/2 - o|) & \text{if } o \neq 2^{n-1} \\ 2^{n-1} & \text{if } o = 2^{n-1} \text{ and } r \in \{1, 256/2^n + 1\} \\ 2^n & \text{if } o = 2^{n-1} \text{ and } r \in \{2, \ldots, 256/2^n\} \end{cases} \tag{4.3}$$

The explanation of these results is analogous to the explanation of the results in the case of an aligned S-Box. Note that in the case of an even offset, the offset is a multiple, but not necessarily a power of two.

We start with the explanation of the offset $o = 2^{n-1}$, i.e., $o = w/2$. Specifically, we start with $o = 2^{n-1}$ and $r \in \{2, \ldots, 256/2^n\}$, which is the last case of Equation 4.3. The reason for this result is the same as for an aligned S-Box. The values which can cause an access to one of these middle rows are consecutive and differ in their $n + 1$ lsb. From these $n + 1$ lsb, however, the two msb can only attain two different values, such as `0b01` and `0b10`. Consequently, there are only $2^{n+1-1} = 2^n$ candidates for the $n + 1$ lsb, which are all covered by the state bytes which can lead to an access to that row. As it is the case for an aligned S-Box, all sets of candidates consist of the same 2^n candidates. Hence, 2^n candidates per key byte remain.

In the case that the S-Box exhibits the offset $o = 2^{n-1} = w/2$ and the attacker observes accesses to the first or the last row, the argument is again the same: in the first and the last row, the input elements to the S-Box are consecutive from 0 to $2^{n-1} - 1$ or from $255 - 2^{n-1} + 1$ to 255. Thus, they differ only in their $n - 1$ least significant bits. Since there are only 2^{n-1} input elements to these rows, they cover all 2^{n-1} possibilities for the given $2^n - (n - 1)$ msb. Hence, the same argument as for an aligned S-Box applies, albeit not for 2^n, but for 2^{n-1}. Consequently, 2^{n-1} candidates remain.

Now, we discuss the first case, which is the best case from an attacker's perspective in the case of an S-Box with an even offset. Again, the state bytes which can cause an access to the observed row are consecutive, and their n lsb cover all possibilities. However, due to the offset, the elements which can cause such an access differ not only in their least significant n bits, but in their least significant $n + 1$ bits. There are three possibilities for the 2 msb of these bits, such as `0b01`, `0b10`, and `0b11`. The middle $2^{n-1} = w/2$ input bytes to the selected row have one of these values, $|w/2 - o|$ have another one, and the remaining $w/2 - |w/2 - o|$ have a third one. Consequently, the plaintext bytes which lead to an access to the observed row, i.e., the elements of the set Z, have the same segmentation, but in different orders, and so has each candidate set $\widetilde{\mathcal{K}}^0_{m_i}$. Interestingly, not only three, but all four values for these two bits, i.e., `0b00`, `0b01`, `0b10`, and `0b11` are attained when we look at the union of all candidate sets. For three of these values, however, there are at least two plaintext bytes whose corresponding candidate sets do not contain any element with the respective values for these two bits. Thus, all elements in the intersection of all 2^n sets of candidates differ only in their $n - 1$ lsb and the intersection cannot have more than $2^{n-1} = w/2$ elements. However, not all candidate sets contain $2^{n-1} = w/2$ elements with the correct values for the relevant two middle bits. Each candidate set contains as many elements with the correct values for these two bits as there are plaintexts in Z which have the same two bit values. Since the plaintexts follow the same segmentation as the row elements, there are candidate sets with only $\min(|w/2 - o|, w/2 - |w/2 - o|)$ elements with the correct bit values. Hence, the intersection can only contain $\min(|w/2 - o|, w/2 - |w/2 - o|)$ elements. On the other hand, it does not contain less than $\min(|w/2 - o|, w/2 - |w/2 - o|)$ elements. In the case of an even offset, the candidates occur in groups of $\min(|w/2 - o|, w/2 - |w/2 - o|)$ elements. These groups remain constant even if another row is observed and also when another key is used. Hence, each set of candidates comprises the same $\min(|w/2 - o|, w/2 - |w/2 - o|)$ elements.

For $n \in \{3, 4\}$, it holds that $\min(|w/2 - o|, w/2 - |w/2 - o|) = 2^i$ if $2^i \mid o$ and $2^{i+1} \nmid o$ for $1 \leq i \leq n - 2$. Thus, the attacker is left with 2^i candidates per key byte in this case.

Example 4.2. *As in Example 4.1, we assume a memory line width $w = 8$ and we observe row $r = 12$. Let the S-Box exhibit an offset $o = 6$. Thus, we have $o = 2^i \cdot 3$ for $i = 1$. For the given parameters, it is $R_{12} := \{82, \ldots, 89\} = \{$`0x52`$, \ldots,$ `0x59`$\}$. The*

Input	Key Candidates
0x60	**0x32** **0x33** 0x34 0x35 0x36 0x37 0x38 0x39
0x61	**0x32** **0x33** 0x34 0x35 0x36 0x37 0x38 0x39
0x64	0x30 0x31 **0x32** **0x33** 0x36 0x37 0x3c 0x3d
0x65	0x30 0x31 **0x32** **0x33** 0x36 0x37 0x3c 0x3d
0x66	0x30 0x31 **0x32** **0x33** 0x34 0x35 0x3e 0x3f
0x67	0x30 0x31 **0x32** **0x33** 0x34 0x35 0x3e 0x3f
0x6a	**0x32** **0x33** 0x38 0x39 0x3c 0x3d 0x3e 0x3f
0x6b	**0x32** **0x33** 0x38 0x39 0x3c 0x3d 0x3e 0x3f

Table 4.2: Example of an SPEA for the first key byte of an AES implementation with an S-Box of width 8 with offset 6. The left column shows the plaintext bytes which led to an access to row 12 of the S-Box. The further entries of the lines show the resulting key candidates. The correct key byte is 0x32, which is one of the two candidates belonging to all eight sets. These two candidates are printed in bold.

first access is observed for the plaintext byte 0x60*. This leads to eight key candidates:* $\widetilde{\mathcal{K}}^0_{0x60} = \{0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39\}$*, since* $(0x60 \oplus k) \in R_{12}$ *for all* $k \in \widetilde{\mathcal{K}}^0_{0x60}$*. All these candidates have the same* $4 = 2^{n-i}$ *msb* 0b0011*, but differ in their four lsb. Table 4.2 shows the complete example by presenting all key candidates which are derived from the S-Box accesses that have been observed for the input bytes* 0x60, 0x61, 0x64, 0x65, 0x66, 0x67, 0x6a*, and* 0x6b*. There are two candidates which belong to all eight sets* $\widetilde{\mathcal{K}}^0_{0x60}$ *to* $\widetilde{\mathcal{K}}^0_{0x6b}$*,* 0x32 *and* 0x33*. Thus, this key byte cannot be revealed uniquely.*

The concrete values for an attack against an S-Box with an even offset are shown in Table 4.4 for $w = 8$ ($n = 3$) and in Table 4.5 for $w = 16$ ($n = 4$). As can be seen in these tables, in the case of an even offset an attacker should observe the first or the last row. If she knows that the offset is not 2^{n-1}, the choice of the row is of no relevance.

S-Box with Odd Offset For an S-Box with an odd offset, this Simple Photonic Emission Analysis results in a complete key recovery. Independent of the observed row and the concrete offset, only a single key candidate remains per key byte if the S-Box exhibits an odd offset. This is the correct key byte.

The explanation for this case is similar to the explanation for the case of an even offset different from $w/2$. There, the number of remaining key candidates can not be bigger than $\min(|w/2 - o|, w/2 - |w/2 - o|)$. The same arguments apply to an odd offset, but in this case the number of remaining key candidates is even smaller. From the candidates which have the correct bits at the relevant position, there are exactly $\min(|w/2 - o|, w/2 - |w/2 - o|)$ plaintexts which each have exactly $\min(|w/2 - o|, w/2 - |w/2 - o|)$ candidates with this property. The union of these candidates consists of more than $\min(|w/2 - o|, w/2 - |w/2 - o|)$ candidates, but

Input	Key Candidates
0x60	0x31 **0x32** 0x33 0x34 0x35 0x36 0x37 0x38
0x61	0x30 **0x32** 0x33 0x34 0x35 0x36 0x37 0x39
0x63	0x30 0x31 **0x32** 0x34 0x35 0x36 0x37 0x3b
0x64	0x30 0x31 **0x32** 0x33 0x35 0x36 0x37 0x3c
0x65	0x30 0x31 **0x32** 0x33 0x34 0x36 0x37 0x3d
0x66	0x30 0x31 **0x32** 0x33 0x34 0x35 0x37 0x3e
0x67	0x30 0x31 **0x32** 0x33 0x34 0x35 0x36 0x3f
0x6a	**0x32** 0x38 0x39 0x3b 0x3c 0x3d 0x3e 0x3f

Table 4.3: Example of an SPEA for a single key byte of an AES implementation, given a memory width 8 and an S-Box with odd offset 7. The left column shows the plaintext bytes which led to an access to row 12 of the S-Box. The further entries of the lines show the resulting key candidates. The correct key byte 0x32, which is the only candidate that belongs to all eight sets, is printed in bold.

in each set, another candidate is missing. All but the correct key is missing once. Hence, the number of remaining key candidates is always 1. This is true for all memory line widths which are a power of 2, i.e., $w = 2^n$ for some $n \in \{0, \ldots, 8\}$

Example 4.3. *As in Examples 4.1 and 4.2, we assume a memory line width* $w = 8$ *and we observe row* $r = 12$*. Let the S-Box exhibit an offset* $o = 7$*. In Table 4.1 it is shown how the S-Box is stored for these parameters. Thus, we have* $R_{12} := \{81, \ldots, 88\} = \{\texttt{0x51}, \ldots, \texttt{0x58}\}$*. The first access is observed for the input byte* 0x60*. This leads to eight key candidates:* $\widetilde{\mathcal{K}}^0{}_{0x60} = \{$0x31*,* 0x32*,* 0x33*,* 0x34*,* 0x35*,* 0x36*,* 0x37*,* 0x38$\}$*, since* $\texttt{0x60} \oplus k \in R_{12}$ *forall* $k \in \widetilde{\mathcal{K}}^0{}_{0x60}$*. Table 4.3 shows the complete example by presenting all key candidates which are derived from the S-Box accesses that have been observed for the input bytes* 0x60, 0x61, 0x63, 0x64, 0x65, 0x66, 0x67*, and* 0x6a*. The only key candidate which belongs to all eight sets* $\widetilde{\mathcal{K}}^0{}_{0x60}$ *to* $\widetilde{\mathcal{K}}^0{}_{0x6a}$ *is the value* 0x32*. Thus, we revealed the first key byte uniquely.*

This example corresponds to Table 4.4, which states that odd offsets always result in unique access patterns, allowing for full key recovery. This also holds true for SRAM storing 16 bytes in each row, see Table 4.5. Thus, in this case an attacker can observe an arbitrary row.

From the view of an attacker, and given an aligned S-Box, longer memory rows make the attack less successful. This also holds true for an offset which is a power of 2. For any odd offset, however, the size w of the memory rows is irrelevant. Hence, an S-Box with an odd offset is a particularly promising target for an attacker. Given that the attacker should not be able to influence the choice of memory and the offset, this has consequences for chip designers and the implementations. An S-Box which is secured as good as possible against such an SPEA should be aligned and stored in memory with rows as long as possible.

Offset	Remaining Candidates per Key Byte		Unresolved Bits of the 128-Bit Key	
	$r = 1$ or $r = 33$	$r \in \{2, ..., 32\}$	$r = 1$ or $r = 33$	$r \in \{2, \ldots, 32\}$
0	8	8	48	48
1	**1**	**1**	**0**	**0**
2	2	2	16	16
3	**1**	**1**	**0**	**0**
4	4	8	32	48
5	**1**	**1**	**0**	**0**
6	2	2	16	16
7	**1**	**1**	**0**	**0**

Table 4.4: Number of remaining candidates per key byte and unresolved bits of the full 128-bit key, depending on the offset and row, when each SRAM row stores 8 bytes. For an offset of 0, there are only 32 rows containing S-Box elements.

In our exemplary practical attack both DUAs had a maximum odd offset $o = w - 1$. In the case of the ATMega328P, the S-Box had an offset of $o = 7$ as in Table 4.1. Similarly, the S-Box on the ATXMega128A1 had an offset of $w = 15$, leaving just a single element of the S-Box in the first row of SRAM containing S-Box elements. Since the S-Box exhibited a maximal offset $o = w - 1$ on both DUAs, we set the detector to measure the emissions of the row driver of the first row. Since the first row only retained a single S-Box element, all key bytes k^0 to k^{15} could be directly inferred from the time of the access and from the plaintext byte m_i that caused the access: the time of the access revealed which of the 16 key bytes caused the access, and the corresponding plaintext byte revealed the value of this key byte.

Figures 4.2 and 4.3 show the time-resolved photonic emission traces for both practical attacks. Due to the described attack scenario where only accesses for the state value `0x00` were observed, the colored lines and their annotated byte values both correspond to the input plaintexts that caused a row access and the key candidate which was derived from that access. The equidistance of the 16 peaks follows from the equidistant computation of the `SubBytes` operation for the 16 AES state bytes.

AES-192 and AES-256

The attack can be directly transferred to AES-192 and AES-256, where the attacker wants to reveal the secret key $k \in \{0,1\}^{192}$ and $k \in \{0,1\}^{256}$, respectively. The minimal number of unresolved bits of an AES-192 or AES-256 key is shown in Figures 4.6 and 4.7, respectively.

For the first 16 bytes k^0 to k^{15} of the respective keys, the attack and analysis are the same as for AES-128. For the remaining bytes k^{16} to k^{23} and k^{31}, respectively, the S-Box accesses during the second `SubBytes` operation have to be observed as

Offset	Remaining Candidates per Key Byte		Unresolved Bits of the 128-Bit Key	
	$r = 1$ or $r = 17$	$r \in \{2, ..., 16\}$	$r = 1$ or $r = 17$	$r \in \{2, \ldots, 16\}$
0	16	16	64	64
1	**1**	**1**	**0**	**0**
2	2	2	16	16
3	**1**	**1**	**0**	**0**
4	4	4	32	32
5	**1**	**1**	**0**	**0**
6	2	2	16	16
7	**1**	**1**	**0**	**0**
8	8	16	48	64
9	**1**	**1**	**0**	**0**
10	2	2	16	16
11	**1**	**1**	**0**	**0**
12	4	4	32	32
13	**1**	**1**	**0**	**0**
14	2	2	16	16
15	**1**	**1**	**0**	**0**

Table 4.5: Number of remaining candidates per key byte and unresolved bits of the full 128-bit key, depending on the offset and row, when each SRAM row stores 16 bytes. For an offset of 0, there are only 16 rows containing S-Box elements.

well. For these key bytes, however, the attack presented in Section 4.1.2, using only the described 256 messages m_i, reveals fewer bits per key byte than expected. This can have two different causes.

The first one occurs when the S-Box exhibits either no or an even offset and $x > 1$ candidates remain for each of the key bytes k^0 to k^{15}. It stems from the operation MixColumns, which has been executed once before the second SubBytes operation. During the MixColumns operation, four state bytes are combined. Thus, after the MixColumns operation, each state byte depends on four former state bytes. In case the respective key bytes have not been fully revealed during the first round, however, an attacker does not know these four state bytes exactly, but has x candidates for each of them. Consequently, she has up to x^4 candidates for each state byte after the MixColumns operation. The success of the attack against the first SubBytes operation, however, is based on the exact knowledge of the state bytes before the initial AddRoundKey operation. This is ensured since the attacker chooses the plaintexts, and these are exactly the state bytes before the initial AddRoundKey operation. Since the attacker does not know the value of the state bytes directly before the second AddRoundKey operation in case she does not reveal the first 16 key bytes completely, the observed S-Box accesses have to be related to each of these candidates. Thus, if y candidates remain per key byte during the analysis of the

Offset	unresolved bits of the full AES-192 key	
	r = 1 or r = 33	$r \in \{2, ..., 32\}$
0	72	72
1	0	0
2	24	24
3	0	0
4	48	72
5	0	0
6	24	24
7	0	0

Table 4.6: Minimal number of unresolved bits of an AES-192 key, depending on the offset and row, when each SRAM row stores 8 bytes. For an offset of 0, there are only 32 rows. Attacking AES-192 requires measuring the photonic emissions during the first two SubBytes operations.

leakage in the second round, there are up to $x^4 \cdot y$ candidates altogether for each of the key bytes k^{16} to k^{23} and k^{31}, respectively.

The second effect generally leads to $y > x$: during the attack of the first 16 key bytes, due to the chosen input messages and the bijective AddRoundKey operation, the S-Box gets accessed at 256 pairwise different addresses for each of the 16 bytes, so that all of its elements are accessed exactly once. During the second round, however, the state values leading to the S-Box accesses are not the result of a bitwise XOR of the input plaintext and the secret key. Here, they are the result of the initial AddRoundKey operation and the complete first round, i.e., SubBytes, ShiftRows, MixColumns, and again AddRoundKey. All operations except MixColumns are - observing only one of the 16 state bytes - injective, whereas the MixColumns operation maps always four bytes to one byte by mixing these four bytes. Using just the input messages m_i explained above for the practical SPEA against AES-128, this implies that at the beginning of round two, we do not have 256 pairwise different values per state byte, but considerably less. In fact, we can expect approximately 162 pairwise different accesses. Thus, we observe fewer accesses to a fixed row and consequently, fewer key candidates get ruled out.

The expectation value 162 is calculated as follows: based on the 256 chosen input messages m_i and the MixColumns operation, we define the discrete random variables $X_i, i \in \{1, \ldots, 256\}$. These variables describe the number of pairwise different S-Box inputs in round two when i S-Box accesses have been observed in succession. Note that here, the accesses have not been observed for a specific row, but for the whole S-Box. The i single S-Box accesses are independent from each other and assumed to follow a uniform distribution on $\{0, \ldots, 255\}$. Thus, X_i takes values in $\{1, \ldots, i\}$ and $\mathbb{E}(X_i) = \sum_{j=1}^{i} j \cdot \mathrm{p}(X_i = j)$. The probabilities $\mathrm{p}_{i,j} := \mathrm{p}(X_i = j)$ for $i \in \{1, \ldots, 256\}$, $j \in \{1, \ldots, i\}$ can be defined recursively: for $i = 1$, we have

Offset	unresolved bits of the full AES-256 key	
	r = 1 or r = 33	$r \in \{2, ..., 32\}$
0	96	96
1	0	0
2	32	32
3	0	0
4	64	96
5	0	0
6	32	32
7	0	0

Table 4.7: Minimal number of unresolved bits of an AES-256 key, depending on the offset and row, when each SRAM row stores 8 bytes. For an offset of 0, there are only 32 rows. Attacking AES-256 requires measuring the photonic emissions during the first two SubBytes operations.

$p_{1,1} = 1$. For $i = 2$, we have $p_{2,1} = \frac{1}{256}$ and $p_{2,2} = \frac{255}{256}$. For $i > 2$, there is a case differentiation depending on the value j.

$$p_{i,j} := \begin{cases} p_{i-1,1} \cdot \frac{1}{256} & j = 1 \\ p_{i-1,j-1} \cdot \frac{256-(j-1)}{256} + p_{i-1,j} \cdot \frac{j}{256} & 2 \leq j \leq i-1 \\ p_{i-1,i-1} \cdot \frac{256-(i-1)}{256} & j = i \end{cases}$$

Having computed the probabilities $p_{256,j}$ for all $j \in \{1, \cdots, 256\}$, we can calculate $\mathbb{E}(X_{256}) \approx 162$. The value $\mathbb{E}(X_{256})$ describes the number of pairwise different S-Box inputs that we can expect when we use 256 different input messages. We will always observe 256 S-Box accesses in each round. Since $\mathbb{E}(X_{256}) \approx 162 < 256$, however, we cannot expect to have 256 pairwise different accesses to the S-Box at the beginning of the second SubBytes operation, but only approximately 162. Particularly, we have to expect less than w accesses to the row we are observing. Since each observed access reduces the number of possible keys, more key candidates are left for key bytes k^{16} to k^{23} and k_{31}, respectively, as compared to key bytes k_0 to k_{15}, at least in the case of an S-Box which is not aligned. However, in case an attacker wants to conduct the attack during the first and the second round, she can increase the probability of more pairwise different S-Box accesses during the second SubBytes operation by using more than the 256 input messages m_i defined in Section 4.1.2. Furthermore, it could be beneficial to choose the additional input messages dependent on the revealed key bits to increase the attack success probability.

Other Cryptographic Algorithms

The SPEA presented in this chapter can be easily transferred to other cryptographic algorithms exploiting an S-Box or, more general, precomputed lookup tables. We

sketch the attack for the two lightweight block ciphers PRESENT and Light Encryption Device (LED), although in a risk assessment of lightweight ciphers, side channel attacks do not matter as much as for a stronger cipher [36]. We also explain why SPEA can be used to attack binary exponentiation and multiplication algorithms even if these are protected against timing or power attacks.

PRESENT The ultra-lightweight block cipher PRESENT was published in 2007 [36]. It transforms a 64-bit plaintext in 31 AES-like rounds into a ciphertext. Each round consists of a $\oplus$ operation (addRoundKey), an S-Box lookup (sBoxLayer), and a permutation (pLayer). The key length is 80 or 128 bits, where the developers of PRESENT recommend the 80-bit version for the applications they had in mind, i.e., applications in extremely constrained environments such as Radio-Frequency Identification (RFID) tags. The round key of the first addRoundKey operation consists of the 64 msb of the 80-bit secret key k, i.e., k_{79} to k_{16}. The further round keys are derived from the original secret key mainly by rotation, strengthened by a non-linear substitution of four bits and by flipping up to five bits. Regarding the second round key, its bits 75 to 61 are exactly the 15 lsb of the original secret key, i.e., k_{14} to k_0. The four msb of the second round key are the substitution of the bits k_{18} to k_{15}, i.e., this is where bit k_{15} of the original secret key matters.

The sBoxLayer makes use of a 4-bit S-Box. Thus, the 64-bit state is split into 16 4-bit nibbles, and each nibble is substituted individually, after which the nibbles are concatenated again. This corresponds to the SubBytes operation of AES, except for the bit length of the nibbles. In case the physical attack can be transferred to, e.g., RFID tags, the SPEA from this chapter can be directly translated to the first sBoxLayer of PRESENT, which directly follows the initial addRoundKey operation. Again, depending on the row width and offset, the set of key candidates can be greatly reduced. Since the key length is bigger than the block length, the attacker also has to analyze the second round if she wants to reveal the secret key completely. This does not require additional measurements, but only more memory to store the longer traces. However, as we pointed out for AES, the analysis of the second round might be less efficient.

LED The lightweight block cipher LED was presented in 2011 [84]. It is also based on AES-like design principles. It is a 64-bit block cipher which allows different key sizes, where 64 bits and 128 bits are the most frequent choice. Unlike most other block ciphers, LED does not utilize a key schedule, but uses the original key as it is during each round (for the 64-bit version) or the first and the second half alternating (for the 128-bit version). The state is organized in a 4×4 matrix of 4-bit nibbles. LED consists of an initial key addition ($\oplus$), followed by 8 steps. After each step, again a key addition takes place. Unlike the nomenclature that is used for other block ciphers, each step consists of 4 rounds. These rounds do not use any secret key material. Each round consists of the four operations AddConstants, SubCells, ShiftRows, and MixColumnsSerial. For an SPEA in the manner described above,

only the AddConstants and the SubCells operation of the first round of the first step are relevant. During this AddConstants operation, 8 of the 16 nibbles undergo an $\oplus$ operation with precomputed constants. The eight constants are different, but each summand is known from the description of the cipher. During the subsequent SubCells operation, each nibble is substituted by another 4-bit value which is read from an S-Box. The S-Box which is used in the LED algorithm is the same as used during the sBoxLayer of PRESENT.

Thus, the analytical part of the SPEA on LED differs in two aspects from an SPEA on AES and PRESENT: first, there is not only the key addition before the analyzed substitution operation, but also the AddConstants operation. Second, an attacker who uses the input plaintexts that we used for our attack has to bear in mind that for half of the nibbles, the input to the S-Box is not the $\oplus$ result of the input plaintext and the round key. Nevertheless, since the $\oplus$ operation is bijective if one of the summands is fixed, the attacker will have as many pairwise different S-Box inputs as she uses pairwise different plaintexts.

Both these aspects do not make the attack more difficult, but only slightly increase the effort for the analysis. The attack success again depends only on the offset and the number of columns in the memory where the S-Box is stored. In the case that LED-128 is used, the first SubCells operation of the second step has to be analyzed as well, since otherwise only the first half of the secret key can be revealed. This also slightly increases the effort for the analysis, since until secret key material is used again after the initial key addition, a whole 4-round step is executed.

Binary Exponentiation and Multiplication Algorithms Binary exponentiation and multiplication algorithms have been attacked with timing analyses as well as power analyses. Consequently, countermeasures to equalize their timings and power consumptions for both possible bit values have been developed. However, different bit values do not only lead to different timing and power consumption characteristics in unprotected devices, but the registers on which vulnerable operations are performed also depend upon them. This is not addressed by countermeasures against timing and power attacks, since these do not exploit location-dependent differences. Thus, it was proposed to exploit this leakage in side channel attacks which do exploit location-dependent differences [93]. The authors present a successful attack against an elliptic curve scalar multiplication algorithm and reveal a secret 163-bit scalar by analyzing the location-dependent leakage. Hence, binary exponentiation and multiplication algorithms which use different registers depending on the value of the processed bit of the secret exponent can also be attacked with a Simple Photonic Emission Analysis.

4.1.3 Countermeasures

First, we discuss generic countermeasures which are not recommended to mitigate SPEA. Then we present countermeasures which will at least make a successful attack considerably harder for an experienced attacker. Given that the proposed SPEA

requires the attacker knowing the plaintexts (or the ciphertexts, if the decryption process is attacked), the countermeasures have to be applied especially to the first and the last rounds of block ciphers. When the attacker can only measure the interesting emission during the middle rounds, she does not know the value of the relevant state bytes. This prevents the analysis presented in this chapter.

Ineffective Countermeasures

Whenever successful cryptanalyses against block ciphers are published, there are two generic countermeasures which are often proposed: increase the safety margin of the cipher by using longer keys and by increasing the number of rounds [151]. This demand is useful for many pure cryptanalytical attacks, but it is often not useful for side channel attacks. In case of SPEA, neither longer keys nor more rounds would effectively prevent the exposure of the secret key. The attacks presented in this section show that their success is independent of the number of rounds. They do not show that the attack success is independent of the key length. However, we explained that the attack against AES-128 can be transferred to AES-192 and AES-256 for little cost. Thus, concerning SPEAs on block ciphers, longer keys increase the safety margin only marginally and are therefore not recommended as countermeasure.

For algorithms which utilize a non-linear substitution, the choice of the S-Box is of major importance. Therefore, developers of such algorithms analyze the security of S-Boxes of a fixed size before choosing the one that will eventually be used. The developers of PRESENT say that they first classified all 4-bit S-Boxes according to certain necessary conditions and then chose one which is particularly suitable for constrained hardware implementations [36]. For an SPEA in the manner described above, however, the choice of the S-Box is irrelevant since only the S-Box accesses are observed. The attacker wants to reveal the inputs to the S-Box. Therefore, the cryptanalytic security provided by the S-Box does not matter.

Memory encryption has no effect on the optical emission of memory accesses. For SPEA, the memory access patterns would be unaffected. However, memory encryption would make the initial spatial analysis more cumbersome. It would obfuscate the values in memory, preventing the memory from being read out optically.

Memory scrambling, whose goal is to obfuscate the layout of memory in terms of addresses, would potentially make the attack even easier once the attacker has gained spatial orientation on the DUA. It increases the likelihood that a single S-Box element is isolated in a row of memory. This increases the attack success.

Randomized delays such as Random Process Interrupts (RPIs) or additional `NOP` instructions, could be thwarted by employing an APD detection gate long enough to encompass any and all randomization clock cycles. Since accesses to the S-Box occur at vastly different points in time, the resulting temporal resolution would still be sufficient to yield all S-Box accesses.

It has been argued that shrinking structure sizes will eventually defeat photonic emission analysis. However, recent works show that shrinking feature sizes do not

eliminate optical emission [161, 169, 171]. Moreover, photonic detection techniques continue to improve rapidly. Also, in practice, structure sizes only apply to the smallest structures on an IC. In every IC there are many transistors which have far larger channels than that of the smallest logic on the chip. These also process relevant data. One such example is the row driver exploited in this work, which is sized up to cope with the large capacitances of SRAM rows.

Effective Countermeasures

The concrete SPEA presented in this section could be prevented by calculating the substitution value in $\mathbb{F}_{2^8}$, rather than using a lookup table during the SubBytes operation. However, this would increase the necessary time for encryption and decryption considerably. In addition to this caveat, this would still allow for a Differential Photonic Emission Analysis, see Section 4.2, since the values of the state bytes before and after the `SubBytes` operation would still be stored in the same register directly after each other. Independently, the bitslice AES implementation by Rebeiro et al. [142] should also be immune against this SPEA.

Given that the implementation makes use of an S-Box, the worst case for an attacker is an aligned S-Box, as can be seen in Tables 4.4 and 4.5. In this case, 2^n candidates remain per key byte, for $w = 2^n$ the number of bytes per memory row. For AES-128 and $w = 8$, 48 bits of the secret key remain secret after the attack. Thus, an implementation which forces an aligned S-Box makes the Simple Photonic Emission Analysis considerably less powerful. The longer the memory lines are, the greater is this effect.

Another very effective countermeasure also concerns the implementation. The SPEA presented above is very effective since all substitution operations make use of the same instance of the S-Box. In case there are several instances of the same S-Box used during each round, only a part of the state bytes accesses the same table. Thus, an attacker with only a single detector gains information only about those bytes which access the observed table. Since such an implementation increases the necessary space, however, it cannot be used in space-constrained environments. To accelerate the computation of symmetric cryptography, this method was already described for hardware implementations of symmetric ciphers which employ table lookups [68]. The authors propose an instruction for parallel table lookups as a supplement to a basic Reduced Instruction Set Computer (RISC) processor. For their synthesized PoC AES implementation using this novel instruction, the encryption is indeed faster than without the parallelization. Thus, parallelization also brings further advantages if execution time matters. Interestingly, the authors say that their optimized implementation eliminates the variability of encryption time. This might even strengthen the implementation against timing attacks.

The above proposed countermeasure can also be generalized: under the assumption that an attacker uses only a single detector, it has to be prevented that all necessary information can be captured at a single spot such as a single transistor. Parallelization is one approach to prevent this, but others are possible as well, e.g.,

storing the S-Box at another location after half of the bytes are substituted. In the case that an attacker has multiple detectors, it has to be ensured that these are neither sufficient. Thus, given that she has n detectors, the relevant information should be spread over more than n spots.

Given that the SPEA proposed in this chapter requires the attacker knowing the input bytes, randomized masking techniques as proposed against timing and power analysis attacks might prevent successful attacks [32, 56]. In the case of a photonic emission attack, the randomization does not only obfuscate the mask, but even prevents recording a valid signal due to the need for integrating over thousands of measurements to yield a sufficient SNR.

Another randomization countermeasure proposed against location-dependent side channel leakage can also help to thwart photonic emission attacks. When different registers are used depending on the processed sensitive values, "the assignment of concerned registers to physical locations should be repeatedly randomized [...] during execution" [93]. This is similar to the proposed parallelization countermeasure. Parallelization, however, is suggested to prevent that all necessary information can be captured at a single spot, while randomization of physical locations is suggested when the actual activity of a certain location reveals secret information. Thus, a concrete realization of this randomization countermeasure consists in randomly swapping the S-Box rows during each SubBytes operation, so that an observed row access does not reveal information about the state byte.

Many common hardware countermeasures, in fact, do not prevent photonic emission attacks at all. Shields and meshes generally only protect against attacks from the frontside, and thus do not prevent photonic emission analysis from the backside. As a countermeasure that prohibits any optical emission attacks, we therefore propose an active shield or mesh on the backside of an IC. A single metal layer on the backside can trap all photons generated within the chip. As soon as this is incorporated, it has to be protected against being removed. This can be ensured by active integrity checks.

4.2 Differential Photonic Emission Analysis

The following section explains Differential Photonic Emission Analysis in detail. It is based on [104] and [105]. First, we describe the physical attack which precedes the cryptanalysis. Then, we explain the cryptanalytic details of the univariate DPEA. We apply Difference of Means (DoM), Pearson correlation and the stochastic approach. We discuss their efficiency and the importance of the choice of the distinguisher. Finally, we show why certain cryptographic countermeasures do not prevent a DPEA and outline potential effective countermeasures.

4.2.1 Physical Attack

In contrast to DPA, detailed knowledge about the layout of the DUA is necessary, or at least advantageous, to perform a successful DPEA. The attacker has to find a

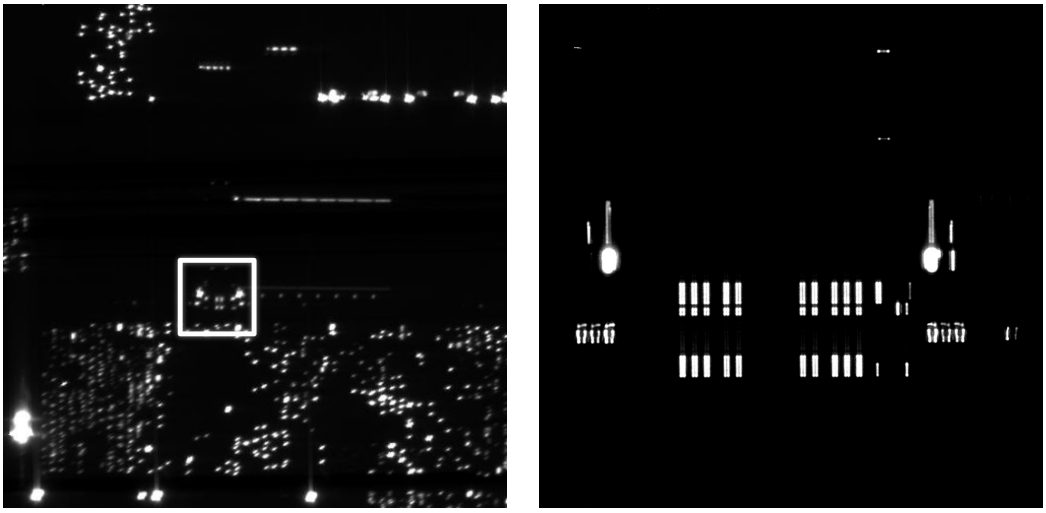
suitable area on the chip before measuring the emissions and performing the cryptanalysis. For this reason it is important to consider several features of the attacked device to fully understand the potential attack surface.

The ATMega328P microcontroller is based on the AVR architecture. The AVR architecture is an 8-bit architecture with a 16-bit or 32-bit fetch and 16-bit data memory addresses. The 8-bit registers `r26` and `r27`, `r28` and `r29`, and `r30` and `r31` form the low and high bytes of 16-bit registers `X`, `Y`, and `Z`, respectively. On the ATMega328P, SRAM is mapped to the data memory and is accessed via load (`ld`) and store (`st`) instructions in conjunction with the registers `X`, `Y`, and `Z` for indirect memory addressing. Load and store operations can also optionally pre- and post-increment or decrement the pointers of the operation. Using one of these pointers makes it particularly easy to perform operations on consecutive bytes of memory. These instructions make it possible to access consecutive bytes of memory without having to reload the pointer. Thus, they help us to target all 16 AES state bytes in a single attack.

The ATMega328P has four 512-byte memory banks. Each bank is individually connected to the rest of the datapath. This connection consists of very large driving inverters. Hence, these represent interesting attack vectors for a DPEA. By studying emission images as explained in Section 3.2.2, we identified these driving inverters. The DPEA thereby targets the AVR architecture's datapath to recover photonic side channel leakage from the fragmentary AES implementation. We determined that the emissions of the datapath's driving inverters are both data and address dependent.

Figure 4.4(a) shows an emission image of the ATMega328P, taken with the Si-CCD during a write operation at address `0x300`. It can be seen that each bank is individually connected to the rest of the datapath. Figure 4.4(b) shows the highlighted area of Figure 4.4(a) in greater detail. While this emission image was taken, the value `0xff` was read. The large driving inverters that connect an SRAM bank to the rest of the data path are clearly visible and it can be seen that the driving inverters for the first and second SRAM bank are mirrored.

For a photonic side channel attack, it is helpful to have more information about the targeted components. It is interesting to know the relation between the inverters and the individual bits they process and represent. We determined the bit order of the inverters by analyzing emission images. The bit order is shown in Figure 4.5(a). Due to the IC's layout, the inverters form two groups, the five msb and the three lsb. Because of the distance between the two groups and the additional enable and clock signals that lie between them, measuring the emissions of both groups in a single trace would imply too much noise. Therefore, we chose to measure the 5 msb and the 3 lsb separately. The position and approximate aperture of the measurements is shown in Figure 4.5(b). We denote the respective sets of measurements with $\mathcal{T}$, i.e., $\mathcal{T}_{msb}$ denotes the set of the traces captured at the chip's msb, and $\mathcal{T}_{lsb}$ accordingly denotes the set of the traces captured at the chip's lsb. To capture all traces, we needed 26 hours with the first setup. Each of these traces consists of 1.000.000 averaged measurements. Since this setup required to be used in gated operation, in

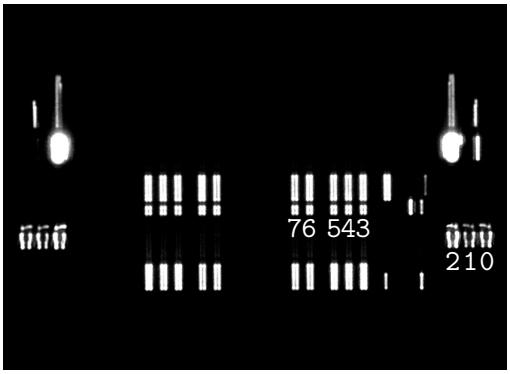


(a) Write operation at address 0x300. (b) Read of the value 0xff.

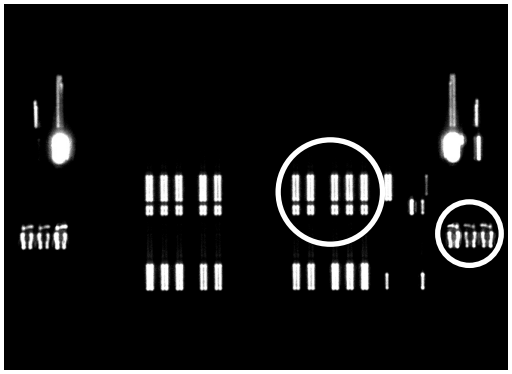
Figure 4.4: Emission images of memory accesses on the ATMega328P. The SRAM line at address 0x300 is clearly visible in Figure 4.4(a). The highlighted area of Figure 4.4(a) is shown in greater detail in Figure 4.4(b). It can be seen that the driving inverters for the first and second SRAM bank are mirrored.

fact even more than 1.000.000 measurements were needed. Given that each trace was composed of 20 measurements, 20.000.000 measurements were necessary for a single trace.

In the subsequent cryptanalysis, we analyzed the emissions of the `SubBytes` operation. The software AES executed on the microcontroller was identical to the implementation employed for the SPEA. Table 4.8 shows the assembly code for the compiled `SubBytes` operation used in the software implementation. Conditional branches, as well as any load or store operations, generally take two clock cycles to execute. Thus, this also holds true for the conditional branching operation `brne` and the load and store operations `ld` and `st`. As we explained in Section 3.2.2, the first element of the S-Box was located at the SRAM address `0x23f`. For each S-Box input, the absolute address of the corresponding S-Box output element consists of the sum (not the $\oplus$ sum) of `0x23f` and the value of the respective state byte. In the `SubBytes` function, register `X` points to the address of the 16 state bytes. To perform the `SubBytes` operation, a state byte is read (`ld r30, X`). The value of this state byte is the result of the initial `AddRoundKey` operation. Next, this value is used to index the AES S-Box by adding the base address of the AES S-Box, i.e., `0x23f`. The S-Box addresses attain values between `0x23f` and`0x33e` since the corresponding state bytes attain values between `0x00` and `0xff`. Thus, to index the first element of the S-Box, its address is `0x23f` + `0x00` = `0x23f`, while the address of the last element is `0x23f` + `0xff` = `0x33e`. The `avr-gcc` compiler uses the subtract operations, `subi`



(a) Bit order of the driving inverters.



(b) Position and aperture of the measurement.

Figure 4.5: Emission images of the driving inverters for the second SRAM bank on the ATMega328P. Figure 4.5(a) shows their bit order. Figure 4.5(b) shows the position and approximate aperture of the measurements.

and `sbci`, and the complementary immediate values `0xC1` and `0xFD` because subtract operations are executed in a single clock cycle. The S-Box output is loaded with the instruction `ld r25, Z`. The 8-bit registers `r30` and `r31` form the low and high bytes of the 16-bit register `Z`. Then, the S-Box output is stored and the `X` pointer is incremented to point to the next state byte. The `cpi` operation ensures that only 16 bytes are actually substituted by the subroutine.

Physical Attack with Improved Setup

In addition to the measurements explained above, we conducted another set of measurements with an improved setup which exceeds the capabilities of the original system by far, see Section 3.2.3. The improved setup allows us to measure the emissions of a single transistor directly. It also allows us to scrutinize the efficiency of the DPEA and different distinguishers in greater detail since with the improved system, each trace is stored separately. As explained above, a single trace cannot be sufficient for an analysis. However, storing each trace separately allows us to determine exactly how many traces are necessary, while the original system only stored accumulated traces from approximately 20.000.000 measurements for each input. Note that the improved setup can be used in free-running mode. The gated operation is no longer necessary. With this setup, we need less than 0.01% of the measurements that we recorded with the first setup. We needed only 11 minutes to capture all necessary traces.

```
1  subBytes:
2  cbi  PORTB, Pin5 ; Set trigger
3  ldi  r24, 0x00   ; i = 0
4  do_subBytes:
5  ld   r30, X      ; Load &state[i]
6  ldi  r31, 0
7  subi r30, 0xC1   ; Add SBox low address byte (0x3F)
8  sbci r31, 0xFD   ; Add SBox high address byte (0x02)
9  ld   r25, Z      ; Load &SBox + &state[i]
10 st   X+, r25     ; Store new state[i]
11 subi r24, 0xFF   ; i++
12 cpi  r24, 16     ; i < 16?
13 brne do_subBytes
14 sbi  PORTB, Pin5 ; Clear trigger
```

Table 4.8: Assembly code of the AES SubBytes Operation.

4.2.2 Cryptanalysis

In this section, we present how the secret key of a cryptographic algorithm can be revealed with a univariate DPEA. We present three different distinguishers: DoM, Pearson correlation and the stochastic approach. We present the analyses in the same order as they were used historically as distinguishers in the side channel literature. All methods show that DPEA is not only a cryptanalytic method, but also helps to gain knowledge about the attacked device. Afterwards, we compare the distinguishers in terms of efficiency and then compare the results hereof with previous results from other side channel attacks.

We attacked AES-128 encryption. As in the SPEA, we attacked and revealed each of the 16 key bytes k^0 to k^{15} separately. Therefore, unless otherwise stated, the description in the remainder of this section always refers to a fixed, but arbitrary byte. As explained above, we used 256 input messages m_i, $i \in \{0, \ldots, 255\}$ with $m_i^j = i$ for all $i \in \{0, \ldots, 255\}$ and $j \in \{0, \ldots, 15\}$. Note that more plaintexts do not increase the attack success if the attacker solely exploits the emissions during the first round. Since the 16 state bytes are processed independently until the SubBytes operation is completed, there exist only $2^8 = 256$ possibilities for each plaintext byte. Thus, we use all possibilities for plaintext values with these input messages m_i.

The analyzed traces were recorded at the driving inverters for the second SRAM bank, see Section 4.2.1. We use the following notation throughout the remainder of this section: we collect 256 traces $\mathbf{t}_i$ of photonic emissions, $i \in \{0, \ldots, 255\}$. The trace $\mathbf{t}_i$ is recorded while the device encrypts the input data m_i with the use of the fixed secret key $k \in \{0,1\}^{128}$. Each trace consists of N points in time, i.e., N is the length of the traces and thus, $\mathbf{t}_i = (t_{i,1}, \ldots, t_{i,N})$. The traces $\mathbf{t}_i$ and their components $t_{i,n}, n \in \mathcal{I} = \{1, \ldots, N\}$, respectively, thus refer to real photonic emissions and each

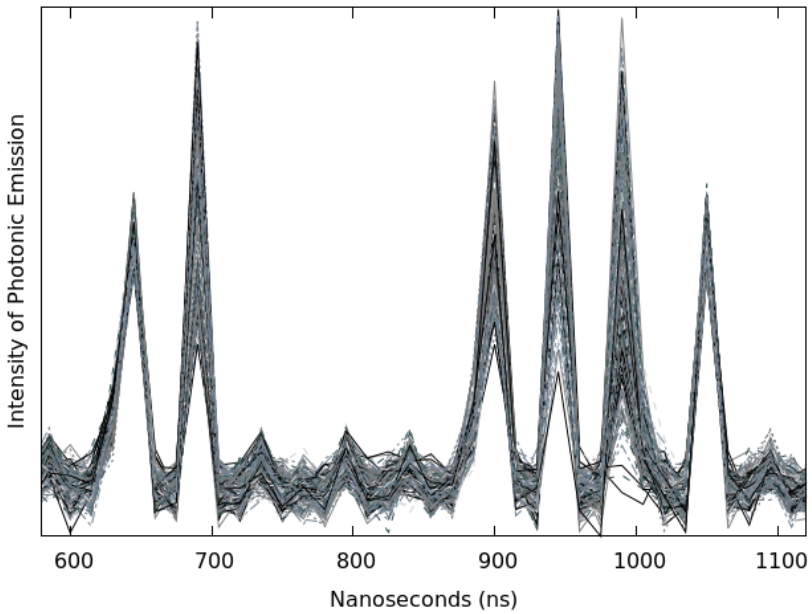


Figure 4.6: Photonic emission traces of the SubBytes operation for a single state byte, captured at the five msb. The three main instructions each take two clock cycles to execute and result in six dominant peaks.

$t_{i,n}$ corresponds to a number of count events. In the following univariate attacks we assume that the emission traces are Gaussian, i.e., they have additive noise and their random parts follow a normal distribution with $\mu = 0$. As stated before, due to the chip's layout we did not measure all eight bits together. Instead, we measured the five msb, i.e., bits b_7 to b_3, and the three lsb, i.e., bits b_2 to b_0, separately, see Figure 4.5(b). The respective sets of traces are denoted with $\mathcal{T}_{msb}$ and $\mathcal{T}_{lsb}$. All traces cover the complete first SubBytes operation, which consists of three main instructions, each taking two clock cycles to execute, as described in Section 4.2.1. These three main instructions are clearly visible as six dominant peaks in Figure 4.6. Since a DPEA requires an intermediate result which depends on the input data and on the secret key, we chose to analyze the second instruction, i.e., the third and fourth of these peaks.

As explained above, we had two sets of measurements: during the first measurements, we averaged one million traces for every input value, in the manner described in Section 3.2.3. This means that due to the gated operation, approximately 20.000.000 computations with the same input were effectively necessary for a single trace. The improved setup allowed us to store each measurement and trace, respectively, separately. With the improved setup, we additionally measured the emissions of individual transistors, especially the one corresponding to bit b_2.

A DPEA reveals the key k of the attacked device by interrelating the captured traces with a hypothesis function h. This hypothesis function models the predicted

location-dependent leakage for the attacked operation and each key candidate based on an assumption about the relation between the cryptographic operation running on the DUA and the side channel leakage. Accordingly, it is also called (leakage) model function [60, 137]. A hypothesis function $h : \mathcal{P} \times \mathcal{K} \times \mathcal{I} \rightarrow Y$ describes potential photonic emissions, based on plaintext byte $m \in \mathcal{P}$, key hypothesis $\tilde{k} \in \mathcal{K}$, and point in time $n \in \mathcal{I} = \{1, \ldots, N\}$. The hypothesis function may, or may not, depend on a given point in time. In case the point in time does not have to be considered, we just write $h(m, \tilde{k})$ instead of $h(m, \tilde{k}, n)$ and omit the third argument. The hypothesis function maps to a discrete image set Y. It may map into the set $Y = \{0, 1\}$, as well as into other sets, e.g., the 9-element set $Y = \{0, 1, \ldots, 8\}$. The latter could be used in case the attacked algorithm operates on bytes, i.e., $\mathcal{P} = \mathcal{K} = \{0, 1, \ldots, 255\}$, and the hypothesis function uses the Hamming Weight (HW) or Hamming Distance (HD) model [112]. The HW of a non-negative number is defined as the number of 1's when the number is written in binary representation. Thus, $HW : \mathbb{N} \rightarrow \mathbb{N}$ with $HW(x) := \sum_{i=0}^{n-1} \mathbb{1}_{\{1\}} x_i$ for $n = \lfloor \log_2 x \rfloor + 1$, where x_i denotes bit i in the binary representation of x. The HD of two non-negative numbers is defined as the HW of the bitwise XOR of these numbers, i.e., $HD : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ with $HD(x, y) := HW(x \oplus y)$.

Difference of Means

The DoM method was not only used in the first published DPA attack [103], but also in the first first published DPEA that led to the recovery of a complete secret cryptographic key [105]. It was also occasionally used for other attacks [112, 118].

The DoM method belongs to the partition distinguishers [162]. It requires and exploits reliable information of just a single bit to reveal the entire key if a nonlinear function can be attacked. The general approach is to partition for each key hypothesis $\tilde{k} \in \mathcal{K}$ the traces according to the value of a certain bit (i.e., 1-bit partition) after a nonlinear function has been calculated. An attacker partitions for each $\tilde{k} \in \mathcal{K}$ all traces according to the value of the chosen bit, which is 0 or 1, respectively. For each $\tilde{k} \in \mathcal{K}$, the attacker thus gets two sets of traces, and calculates a mean for both. Afterwards, she computes the difference of these two mean traces - hence the name, Difference of Means. The underlying assumption is that in case a key candidate is wrong, the partition of the two sets is more or less random, so that both mean traces are approximately equal and thus, the difference trace gets drowned out by the noise. However, in case the traces were partitioned according to the correct secret key k and the emissions of the weighted bit influence the measured photonic signal, there is a significant difference in the two mean traces at some point in time and thus, their difference trace will exhibit a peak at this point.

We applied this distinguisher to the S-Box output of the first round, i.e., $S(m \oplus k)$. Thus, we partitioned for each $\tilde{k} \in \mathcal{K}$ the 256 traces according to the value of one of the bits b_0 to b_7 of the output of the first SubBytes operation. That is, the traces were sorted depending on the value $(\text{SubBytes}\ (m \oplus k))_i$, $i \in \{0, \ldots, 7\}$. We used the lsb measurements $\mathcal{T}_{lsb}$ when we partitioned the traces according to the bits

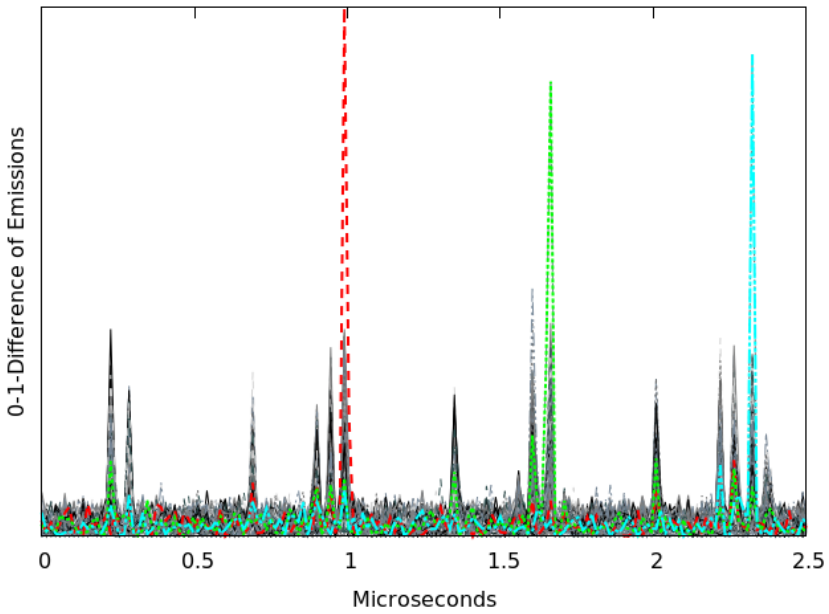


Figure 4.7: Result of a DoM analysis for three key bytes. The msb traces were distinguished based on the value of bit 5. The correct key bytes are plotted in red (dashed), green (dotted), and blue (dash-dotted). All other key candidates are plotted in gray.

0, 1, or 2, and the msb measurements $\mathcal{T}_{msb}$ otherwise. For all analyses, there was no ambiguity when we used the traces for all 256 plaintexts, but we were able to extract the subkeys with a good distinguishability and always yielded $GSR = 1$. Figure 4.7 shows the difference traces restricted to the SubBytes operation of the first three state bytes in the first round when we used all 256 input messages and the corresponding traces from the set $\mathcal{T}_{msb}$. We distinguished them according to the value of the middle bit 5. The explicit equidistant peaks indicate the correct subkeys, which were `0xbd`, `0xdb`, and `0xef` in this attack.

Since the results were unambiguous and could be reproduced also for the other bits, we repeated the same analysis for all bits, but used less than 256 input plaintexts. We could determine the secret key also with fewer plaintexts, as can be seen in Figure 4.8. For this analysis, we used the same sets of traces, but only a random selection of these. The figure shows the relation between the number of pairwise different plaintexts and the achieved min PSR when the emission traces were partitioned according to the value of bits 0, 2, 5, or 6. We repeated every analysis ten times.

The results show that emissions of a single transistor can be sufficient for a successful DPEA. Therefore, we captured another set of measurements with the improved setup. As explained above, we did not store accumulated traces here, but we stored each trace separately. Regarding the three least significant bits, Figure 4.8

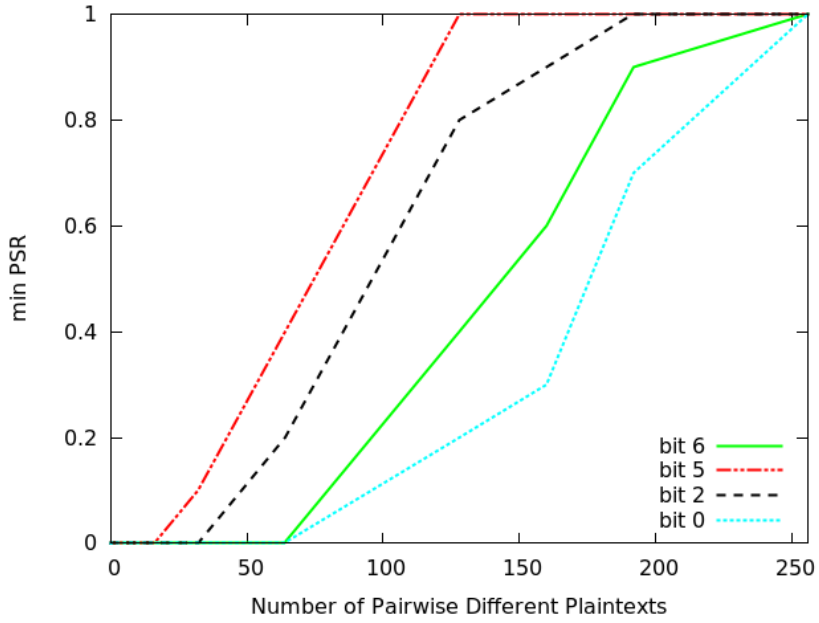


Figure 4.8: Results of a DoM analysis. Relation between the number of pairwise different plaintexts and the achieved min PSR from ten experiments each when 200,000 emission traces were partitioned according to the value of bits 0, 2, 5, or 6.

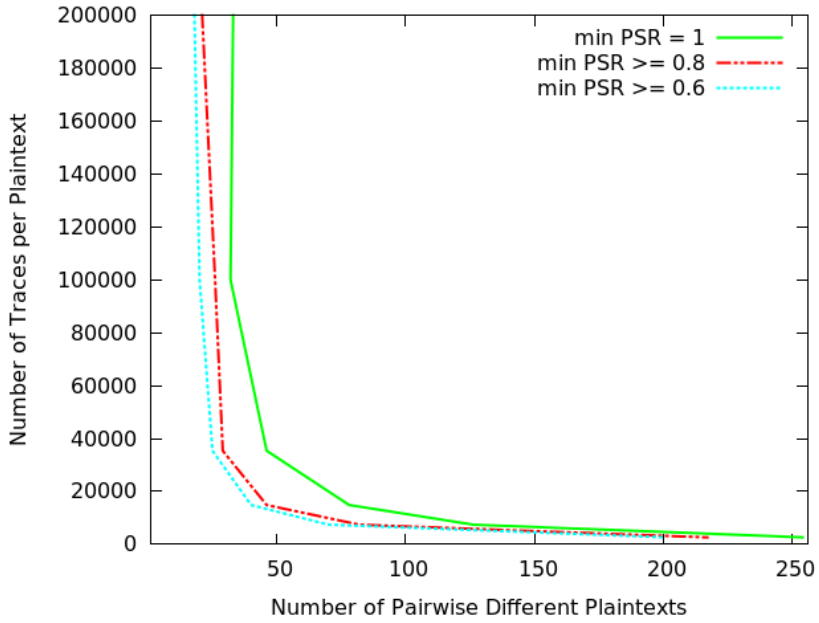


Figure 4.9: DoM analysis with emission traces from a single transistor which corresponds to bit 2. The relation between the number of pairwise different plaintexts and the number of traces per plaintext for a certain min PSR is depicted (min PSR ≥ 0.6 blue (dotted), min PSR ≥ 0.8 red (dash-dotted), min PSR $= 1$ green (solid)).

shows that bit 2 is a good discriminator when the measurements $\mathcal{T}_{lsb}$ from the ATMega328P are analyzed. This result is also in accord with Fig. 4.5(a), where the emissions of bit 2 visibly exceed those of bit 1 or 0. Since bit 2 proved to be a good discriminator, we measured only the emissions of this bit during the second attack. We denote this set of measurements with $\mathcal{T}_{b_2}$. We conducted a DoM analysis with the value of bit 2 as discriminator. Figure 4.9 shows the result of this analysis. The relation between the number of pairwise different plaintexts and the number of traces per plaintext to yield a certain min PSR is depicted (min PSR ≥ 0.6 blue (dotted), min PSR ≥ 0.8 red (dash-dotted), min PSR $= 1$ green (solid)). For each data point, we repeated the analysis 15 times with a random selection of plaintexts. The analysis shows that 200,000 traces per plaintext are not necessary, since the results for this number of traces are equal to the results for 100,000 traces. They are even similar to the results for only 40,000 traces per plaintext. It can be seen that in case the absolute number of traces is restricted, the attacker should use more different plaintexts and fewer traces per plaintext than only a few plaintexts with more traces each. If an attacker uses all 256 possible plaintexts, less than 3,000 traces per plaintexts are necessary to yield min PSR $= 1$. This also implies GSR $= 1$.

Pearson Correlation

The second analysis is strongly related to the DPA using Pearson correlation as statistical distinguisher, which proved to be effective in former side channel attacks [112]. Since we measured two sets $\mathcal{T}_{msb}$ and $\mathcal{T}_{lsb}$ of traces, we also applied this analysis to both sets separately. Thus, indeed we have two independent sets of key candidates, $\mathcal{K}_{msb}$ and $\mathcal{K}_{lsb}$. For simplicity and readability, we refer to these as just $\mathcal{K}$.

Having fixed a certain hypothesis function h and n, one of the points in time, we calculate the correlation coefficient $\rho \in [-1, 1]$ for each key candidate. Here, $\mathbf{t}_n$ denotes the emissions of all considered input plaintexts at time n, i.e., $\mathbf{t}_n$ is a vector whose length equals to the number of considered plaintexts. Given the leakages $\mathbf{t}_n$ and the corresponding hypotheses vectors $\mathbf{h}(\,\cdot\,, \tilde{k})$, the calculation of the coefficient $\rho_{\tilde{k}}$ is done as follows [66].

$$\rho_{\tilde{k}} = \frac{\mathrm{Cov}(\mathbf{t}_n, \mathbf{h}(\,\cdot\,, \tilde{k}))}{\sqrt{\mathrm{Var}(\mathbf{t}_n) \cdot \mathrm{Var}(\mathbf{h}(\,\cdot\,, \tilde{k}))}} \tag{4.4}$$

By calculating the correlation between the hypothesis function vector $\mathbf{h}(\,\cdot\,, \tilde{k})$ and the actual leakage, the attacker determines the unknown key k by selecting the value $\tilde{k} \in \mathcal{K}$ with maximum correlation coefficient ρ. On condition that the hypothesis function is reasonable, wrong key candidates will lead to low correlations, whereas the correct key leads to the highest correlation and reveals itself.

Considering the hypothesis function h, we followed several approaches: we developed hypotheses based on the HW of the processed data, the HD of the pro-

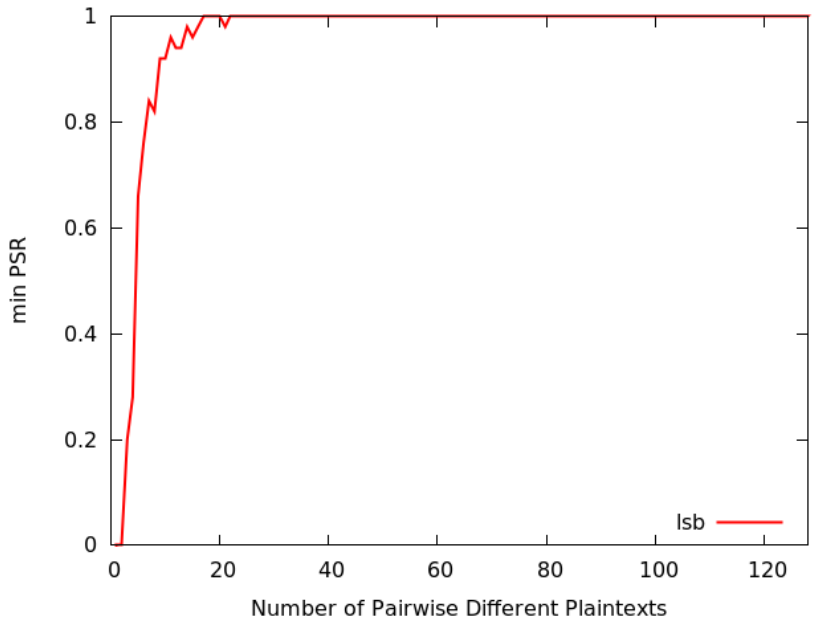


Figure 4.10: Pearson correlation analysis with 200,000 traces per plaintext. Relation between the number of pairwise different plaintexts and the achieved min PSR when the emission traces were analyzed according to the HW of the S-Box input.

cessed data and the result of the preceding suboperation, and the HD of the address of the state byte and the absolute S-Box address. The state bytes were stored from address `0x833` to `0x842` during the first measurements and from address `0x128` to `0x137` during the measurements with the improved setup. Also, we considered all possibilities of incorporating only certain bits, such as including only bits b_0 and b_1, or only b_1, into the hypothesis function for the lsb traces. We also tried out different weighting factors for different bits, including negative weights.

We tested all these functions for both the msb and the lsb measurements against several sets of data, recorded with different secret keys. From all these possibilities, the HW of the S-Box input, i.e., the hypothesis function

$$h(m, \tilde{k}) = HW(m \oplus \tilde{k}), \tag{4.5}$$

proved to be the best hypothesis. We analyzed both sets $\mathcal{T}_{msb}$ and $\mathcal{T}_{lsb}$, recorded with the old as well as the new setup, using Pearson correlation based on the S-Box input as distinguisher. We revealed the full 128-bit secret key without any ambiguity. Figure 4.10 shows the results for the set $\mathcal{T}_{lsb}$, recorded with the improved setup. It shows the relation between the min PSR and the number of pairwise different plaintexts which have been used in the analysis. 200,000 traces per plaintext were used. It can be seen that it is not necessary to use all 256 possible plaintexts, since also 30 pairwise different plaintexts lead to a stable min PSR = 1.

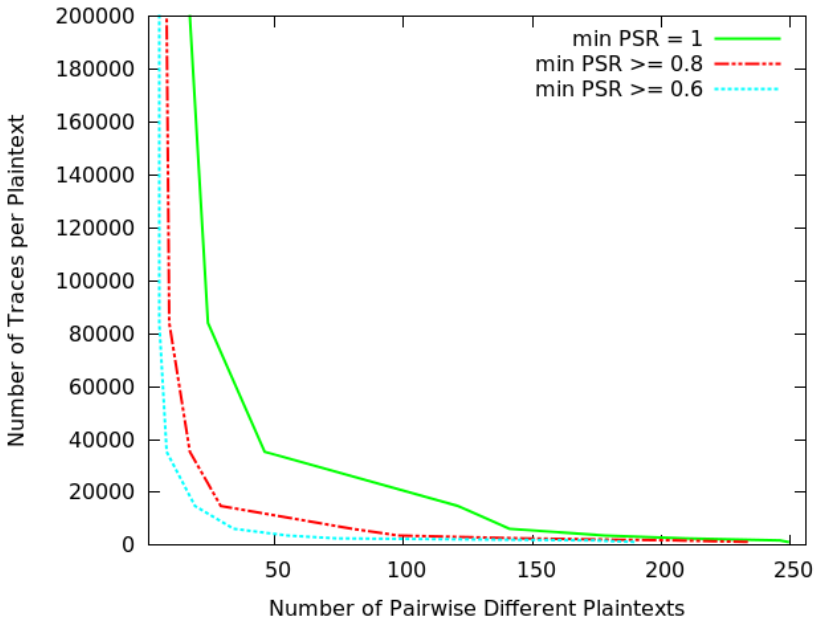


Figure 4.11: Result of the Pearson correlation analysis of the lsb measurements with different numbers of plaintexts and different numbers of traces per plaintext. The achieved min PSR is depicted (min PSR $\geq$ 0.6 blue (dotted), min PSR $\geq$ 0.8 red (dash-dotted), min PSR = 1 green (solid)).

Again, we repeated the same analysis with fewer traces per plaintext. Figure 4.11 shows the results of this analysis for the lsb measurements. We used the same set of traces $\mathcal{T}_{lsb}$ as before, but included only a random selection of these into the analysis. Each analysis was repeated 15 times. Figure 4.11 depicts the relation between the number of pairwise different plaintexts and the number of traces per plaintext to yield a certain min PSR (min PSR $\geq$ 0.6 blue (dotted), min PSR $\geq$ 0.8 red (dash-dotted), min PSR = 1 green (solid)). It can be seen that it is not necessary to use 200,000 traces, since the results for this number of traces are approximately equal to the results for 100,000 traces. For min PSR $\geq$ 0.6 and min PSR $\geq$ 0.8, they are even similar to the results for only 40,000 traces per plaintext. If the absolute number of traces is restricted, more different plaintexts and fewer traces per plaintext lead to better results than than only a few plaintexts with more traces each. If an attacker uses all 256 possible plaintexts, less than 1,500 traces per plaintext are necessary for a stable min PSR = 1. With the first measurement setup, we used approximately 20,000,000 measurements per plaintexts. Thus, we need less than 0.01% of the traces with the improved setup, which is an efficiency increase of a factor of more than 10,000.

Stochastic Approach

We also analyzed the traces with the stochastic approach. The stochastic approach, which was introduced in [148], approximates the real leakage of the exploited side channel as a linear combination of basis functions in a suitable vector space. The stochastic approach can be applied in two scenarios: the stochastic approach with profiling requires additional traces from the DUA, measured while the device encrypts data with another secret key, different from k. Another variant of the stochastic approach can handle scenarios without profiling. Schindler et al. first mentioned that the stochastic approach also allows key extraction without explicit profiling [147]. Doget et al. substantiated this assumption and gave an explicit description, referred to as linear regression [60]. We applied the latter approach to the analysis of photonic emissions to have the ability of a fair comparison of all three used distinguishers. Again, an attacker reveals the full AES key byte by byte when she applies the stochastic approach.

For all key candidates $\tilde{k} \in \mathcal{K} = \{0,1\}^8$ and plaintext bytes $m \in \mathcal{P} = \{0,1\}^8$ we define the function $h\colon \mathcal{P} \times \mathcal{K} \to \mathbb{R}$. Contrary to the hypothesis function used for Pearson correlation, this function h does not describe a hypothesis, but the real data- and key-dependent leakage captured in the traces of photonic emissions. However, the stochastic approach does not aim at the unknown vectors $\mathbf{h}(\,\cdot\,,\tilde{k})$, but at their best approximators $\mathbf{h}^*(\,\cdot\,,\tilde{k})$ in the vector subspaces $\mathcal{F}_{u,\tilde{k}}$, which each are spanned by u basis functions $g_0,\ldots,g_{u-1}\colon \mathcal{P}\times\ \tilde{k} \to \mathbb{R}$ with $g_0(\,\cdot\,,\tilde{k}) = 1$ for all $\tilde{k} \in \mathcal{K}$. These basis functions are selected by the attacker with regard to a sound leakage model. Therefore it is beneficial for an attacker to have knowledge about the attacked device and the implementation of the attacked algorithm. For each key candidate $\tilde{k} \in \mathcal{K}$, the information leakage is then estimated with

$$\mathbf{h}^*(\,\cdot\,,\ \tilde{k}) = \sum_{j=0}^{u-1} \beta^*_{j,\tilde{k}} \cdot \mathbf{g}_j(\,\cdot\,,\tilde{k}). \tag{4.6}$$

An attacker knows the values of the basis functions $g_j(\,\cdot\,,\tilde{k})$, but has to find suitable coefficients $\beta^*_{j,k} \in \mathbb{R}$. To estimate these coefficients, she builds a real-valued matrix $M_{\tilde{k}}$, the dimension of which is determined by the number N of plaintexts which are used in the analysis and the number u of basis functions.

$$M_{\tilde{k}} := \begin{pmatrix} 1 & g_1(m_1,\tilde{k}) & \ldots & g_{u-1}(m_1,\tilde{k}) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & g_1(m_N,\tilde{k}) & \ldots & g_{u-1}(m_N,\tilde{k}) \end{pmatrix}. \tag{4.7}$$

The square matrix $M^T_{\tilde{k}} M_{\tilde{k}}$ is generally regular. Then, the attacker has to find the unique solution $\tilde{\boldsymbol{\beta}}^{\,*} := (\beta^{\,*}_{0,\tilde{k}},\ldots,\beta^{\,*}_{u-1,\tilde{k}})$ for the equation

$$M^T_{\tilde{k}} M_{\tilde{k}} \tilde{\boldsymbol{\beta}} = M^T_{\tilde{k}} \mathbf{t}_n. \tag{4.8}$$

Again, $\mathbf{t}_n$ denotes the emissions of all considered input plaintexts at time n. Afterwards, the proximity between $\mathbf{h}^*(\cdot, \tilde{k})$ and $\mathbf{t}_n$ according to a chosen distance function is calculated for all key candidates $\tilde{k} \in \mathcal{K}$. The least square distance is a sound distance choice [60]. All estimated functions $h^*(\cdot, \tilde{k})$ with $\tilde{k} \in \mathcal{K} \setminus k$ are not related to the deterministic part of the photonic emissions. Hence, it is to be expected that the subspace $\mathcal{F}_{u,k}$ for the correct key k is closer to $\mathbf{h}(\cdot, k)$ than $\mathcal{F}_{u,\tilde{k}}$ is for all $\tilde{k} \in \mathcal{K} \setminus k$. Consequently, we decide for that $\tilde{k} \in \mathcal{K}$ that minimizes $\|\mathbf{t}_n(\cdot) - \mathbf{h}^*(\cdot, \tilde{k})\|^2$ among all candidates and all time instants n within the interval in which the respective key byte is processed.

For this DPEA, the vector subspace $\mathcal{F}_{u,\tilde{k}}$ for $u = 9$ is spanned by $g_0(m, \tilde{k}) = 1$ and $g_j(m, \tilde{k}) := S(m \oplus \tilde{k})_{8-j}$ for $m \in \mathcal{P}$ and $j = 1, \ldots, 8$. We used the stochastic approach without profiling as distinguisher.

Figure 4.12 shows the number of necessary traces per plaintext according to the GSR, which is mapped on the y-axis and was computed on the basis of 32 experiments, which were performed on all 256 plaintexts. For this analysis, both sets $\mathcal{T}_{lsb}$ and $\mathcal{T}_{msb}$ were combined. It can be seen that less than 20,000 traces per plaintext are sufficient to yield a $GSR = 1$.

Figure 4.13 shows the relation between the number of pairwise different plaintexts and the GSR, which is mapped on the y-axis and was computed on the basis of 32 experiments, which were performed with 200,000 traces from both sets $\mathcal{T}_{lsb}$ and $\mathcal{T}_{msb}$ combined. It can be seen that just above 20 plaintexts are sufficient to yield a stable $GSR = 1$ in the case that the emission traces carry a strong signal.

The stochastic approach can also be applied with a profiling phase like in template attacks. Moreover, the underlying distribution of the leakage can be considered, and especially the Poisson distribution promises to increase the attack success. Thus, these results from the stochastic approach are only preliminary results, and it is to be expected that the results will improve in the future.

Comparison and Evaluation

To compare the efficiency of the employed distinguishers, Figure 4.14 combines the data shown in Figures 4.9 and 4.11. For the DoM analysis, the traces $\mathcal{T}_{b_2}$ were used, while for the Pearson correlation analysis, we used $\mathcal{T}_{lsb}$. Each analysis was repeated 15 times. The figure shows the relation between the number of pairwise different plaintexts and the number of traces per plaintext to yield a certain min PSR. For better readability and since the values for min $PSR \geq 0.6$ and min $PSR \geq 0.8$ differ only slightly, we show only the values for min $PSR \geq 0.8$ and min $PSR = 1$ in Figure 4.14. The lines relating to min $PSR \geq 0.8$ are plotted in red (dark), while the lines for min $PSR = 1$ are plotted in green (light). Both lines for the DoM results are dashed, while the lines for the Pearson results are solid. It can be seen that Pearson correlation outperforms the DoM distinguisher. However, the presented Pearson correlation based on the S-Box input reveals only the lsb when the traces $\mathcal{T}_{lsb}$ are analyzed, while the DoM analysis reveals the full secret key using the traces $\mathcal{T}_{b_2}$.

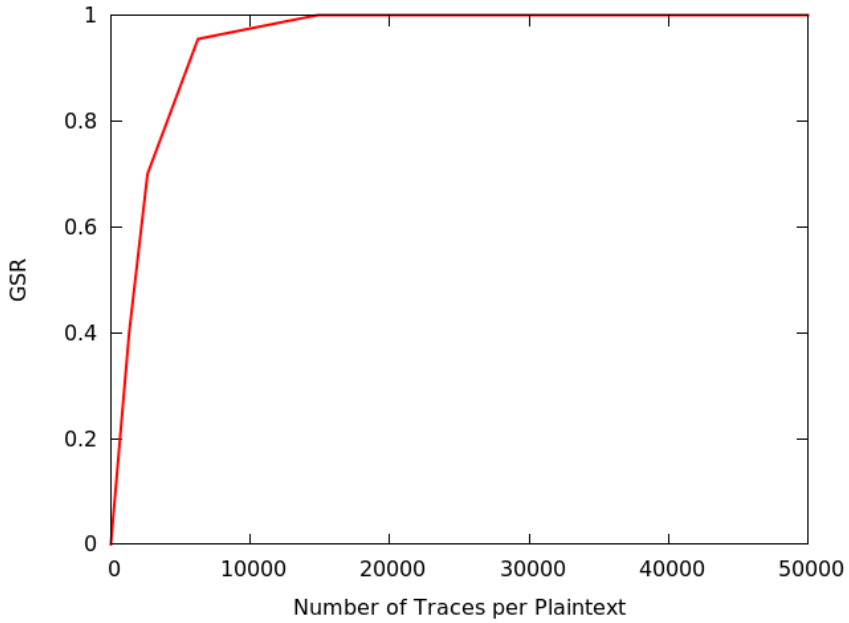


Figure 4.12: Relation between the number of traces per plaintext and the achieved GSR when the stochastic approach is applied to both sets $\mathcal{T}_{lsb}$ and $\mathcal{T}_{msb}$ combined for 256 plaintexts.

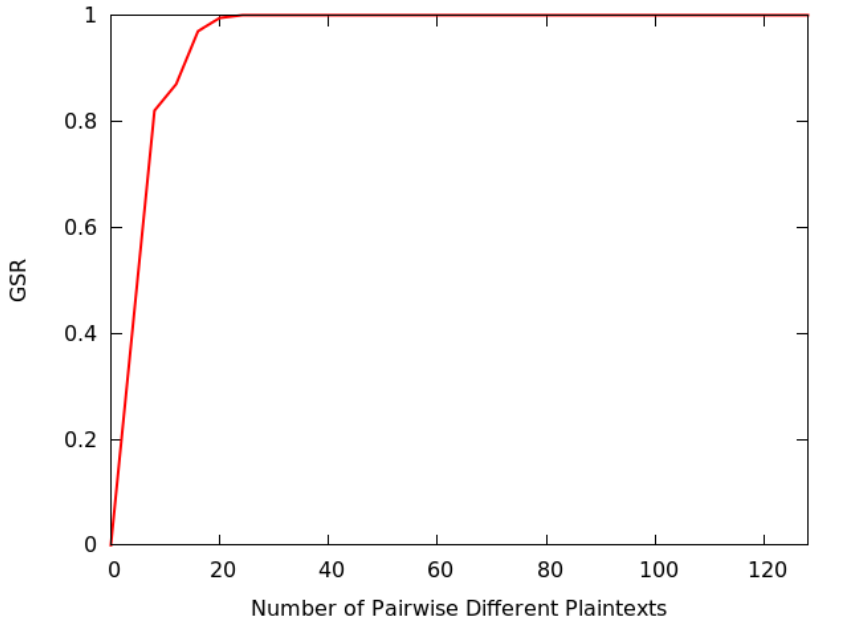


Figure 4.13: Relation between the number of pairwise different plaintexts and the achieved GSR when the stochastic approach is applied to both sets $\mathcal{T}_{lsb}$ and $\mathcal{T}_{msb}$ with 200,000 traces per plaintext.

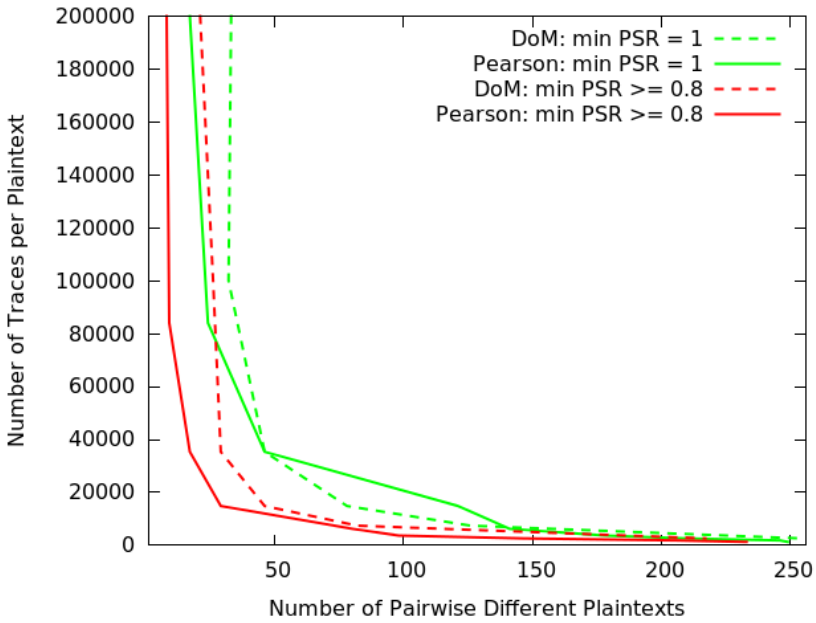


Figure 4.14: Comparison between Pearson correlation and DoM for different numbers of plaintexts and different numbers of traces per plaintext.

The preliminary results for the stochastic approach show that this distinguisher is not worse than the other two distinguishers. Note that min $PSR = 1$ equates to $GSR = 1$. Hence, Figure 4.13 shows that just above 20 plaintexts were needed to yield a stable min $PSR = 1$ when the stochastic approach is used to analyze 200,000 traces per plaintext. This value is similar to the respective values for DoM and Pearson correlation. The stochastic approach might even be more efficient. This is especially true considering that the stochastic approach can not only be used as a generic distinguisher, but can be enhanced with a profiling phase. This has not been used yet. It can also be enhanced by regarding the distribution of the side channel leakage. Since photons emerge following a Poisson distribution, this promises to be a fruitful approach to exploit this leakage property.

The presented analyses lead to slightly different leakage characteristics for the applied distinguishers and altogether, the efficiency of the three distinguishers differs only slightly. These results confirm recent publications on power analysis which show that univariate side channel attacks based on different statistical distinguishers which exploit the same leakage model are asymptotically equal [60, 113].

4.2.3 Countermeasures

First, we discuss generic countermeasures which are not recommended to prevent DPEA. Then, we present countermeasures which will at least make a successful attack considerably harder for an experienced attacker. The countermeasures dis-

cussed for SPEA, such as masking and randomization, have also to be taken into account when considering how differential attacks can be prevented.

Ineffective Countermeasures

Concerning AES, longer keys and more rounds increase the safety margin only marginally and are therefore not recommended as countermeasure. The reason for this is the same as for the SPEA: the secret AES key can be directly revealed during the first one or two rounds, depending on its length. Hence, longer keys and more rounds most probably neither increase the safety margin of other block ciphers which use the whole original secret key at the beginning of an encryption. Block ciphers that use a longer inner state should neither be immune against a DPEA. Even if the time for the statistical analysis increases, this is not critical for the attack success, since the measurement time dominates the overall execution time of a DPEA. Longer inner states would presumably require more plaintexts and hence a longer measurement time. However, given that the experimental setup improved and our latest experiments show that a successful attack against AES-128 does not require many traces as compared to our first experiments, longer inner states should not thwart a DPEA.

Hardware countermeasures include the distribution of bits that compose the same byte over different components of the SRAM, preventing a joint signal from them with a good SNR. However, as soon as an attacker finds out the relevant transistors, DPEA attacks can be applied to measurements of even a single transistor. Hence, this mainly increases the workload to get spatial orientation but also allows for a DoM analysis.

Since the photonic emission is not captured as a global signal, but with a high spatial resolution, dummy operations which are performed at distant spots on the chip that are not under observation do not disturb the photonic signal in the captured traces.

Effective Countermeasures

A common hardware defense against DPA is the introduction of RPIs [54]. Similarly, `NOPs` can be introduced as dummy operations in software to yield the same effect. Intelligent trace alignment, however, makes this countermeasure less effective if an attacker conducting a DPA is not too restricted in the number of traces she can capture [51]. Regarding DPEA, however, it has to be considered that a single trace is not shaped like a power trace, i.e., there is no continuous function with peaks, but only up to a few photons per trace. Thus, the trace alignment is presumably more difficult and hence, introducing RPIs or additional `NOPs` might be an option to prevent DPEA, or at least make it considerably more difficult.

Not only RPIs, but also further properly implemented randomized countermeasures already known for DPA might prevent DPEA at the present time, since DPEA requires averaging at least several hundreds of measurements with the same

data for a single trace. Regardless of whether or not the trace alignment is an easy task, any form of randomization forces longer measurement iteration times, thus greatly increasing trace acquisition times. However, the lifecycle of an IC generation has to be considered, since advances in optical technologies will lead to better measurement tools, which will help attackers to circumvent such countermeasures.

As we explained for the SPEA, it should be prevented that the whole relevant information can be captured at a single spot on the chip. Thus, the advantage that the photonic side channel gives an attacker - the high spatial resolution - should be exploited by chip and software developers by preventing that the captured traces can carry all relevant information and have a good SNR at the same time. When important information is not transferred via the same transistors, an attacker has to have more detectors or to repeat all measurements at several spots when she wants to reveal the full key. Thus, the workload for the attack increases considerably. Without being intended as an active defense, the strong clock signal on the DUA, see Figures 4.4 and 4.5, forced us to measure the emissions for the DPEA twice - once for the msb and once for the lsb. A similar effect should be reached intentionally.

If we had measured all eight transistors jointly, the photonic emissions from the strong clock signal would have hidden the emissions we were interested in. The same effect can be reached by strong light sources close to chip positions where sensitive data are transferred.

Another hardware countermeasure consists in building p-type transistors bigger than n-type transistors. This is already implemented in certain logic components to balance the timing of the different transistors. It would also obfuscate differences in the photonic emissions of 0-1 and 1-0 transitions and therefore prevent more sophisticated analyses as suggested in Section 6.1.

As we explained for the SPEA, an active shield or mesh on the backside of the IC might prevent photonic emission analysis completely.

Chapter 5

Higher-Order Fault Attacks against Pairing Computations

If the adversary can inject multiple faults [...], then an attack could be launched. This however, is an unrealistic attack scenario.

(C. Whelan, M. Scott [180])

The second example of a physical attack which has not been regarded as a realistic threat due to its physical attack complexity is a fault attack against pairing-based cryptography, in particular a higher-order fault attack.

In the field of PBC, higher-order attacks are not only more sophisticated attacks, but actually necessary since, according to current knowledge, a single fault cannot lead to the revelation of the secret argument: a cryptanalyst who wants to reveal the secret input of a pairing computation has to solve both the exponentiation inversion and the Miller inversion, see Sections 2.1.2 and 2.2.2. However, both exponentiation inversion and Miller inversion cannot be solved in polynomial time purely mathematically to date, since they involve solving a polynomial equation of exponential degree [72, 175]. In rare cases like [133], the final exponentiation can be efficiently inverted but in general, both steps are considered to be hard to invert [72, 97]. Therefore, an attacker who has physical access to the DUA might induce faults in the computation to facilitate these problems. Since these faults have to affect both exponentiation inversion and Miller inversion, an attacker has to induce at least two faults to be successful. Then, due to these faults, the attacker does not have to solve the original inversion problems, but easier mathematical functions with lower degree.

Several theoretical fault attacks against PBC have been published in recent years. Most of them focus on the Miller Algorithm, while lately also an attack against the final exponentiation was published. The first fault attack on PBC was presented in 2004 [132]. Page and Vercauteren describe the effect of a modified Miller bound n in the case that the Duursma-Lee algorithm is used for the computation of the pairing. The single-fault attack is applied to several computations of the same pairing until the attacker obtains two computations whose loop bounds differ exactly by one. They also describe the consequences for the exponentiation inversion. In 2006, the same authors published a modified version of their first work and

applied their idea for the fault attack also to the Kwon-BGOS algorithm which is an extension of the Duursma-Lee algorithm to characteristic 2 [133]. In 2007, Whelan and Scott emphasize the importance of the final exponentiation when considering fault attacks on pairings [180]. They describe a data corruption fault on a variant of the η_n pairing and sign change fault attacks on the Weil pairing and on the Tate pairing. They conclude that "pairings with either no or a straightforward final exponentiation are less secure than pairings with a more complex final exponentiation when considering such fault attacks". In 2009, the weakness of the Miller Algorithm in the presence of fault attacks was again analyzed [63]. It was explained that the attacks can be applied when affine, projective, Jacobian, or Edwards coordinates are used. Again, the Miller loop bound was the target of the attack. In 2011, the security of existing point blinding countermeasures against fault attacks on PBC was examined [79]. The authors propose a new countermeasure tailored to the fault attack which aims at having two modified loop bounds which differ exactly by 1. This countermeasure consists in blinding the loop bound, since in many attacks that target this bound, the attacker has to learn the value of the modified loop bound by an additional side channel analysis. In the same work, pairing computations in Edwards coordinates are analyzed with respect to their security related to fault attacks. In 2012, the state of the art of fault attacks against PBC was described in a comprehensive collection of fault analysis in cryptography [65]. In 2013, a novel fault attack that modifies the loop bound by skipping the final `if` instruction was published [13]. The authors do not explain how the exponentiation inversion can be solved. Recently, the first part of the pairing inversion - the exponentiation inversion - was targeted in a fault attack [109]. The authors show how the output of the Miller Algorithm can be computed with at least three faulty computations, each of which requires a special fault. The attack targets the optimized pairing implementation from [154], where the final exponentiation is computed in three steps. Recently, a review of tampering attacks in PBC appeared [34]. Both known side channel attacks and fault attacks on PBC were presented.

Interestingly, none of the proposed attacks have been practically evaluated before we started our research. Till then, all attacks were only described theoretically. We accomplished the task to practically conduct a second-order fault attack and thus proved that fault attacks against PBC are indeed possible and even practical - thus posing a serious threat to cryptographic devices and their sensitive secrets. Our attack benefited from an existing detailed study of the effects of clock glitches on a microcontroller from the AVR family [15]. Simultaneously to our publication [31], similar results were published [110]. However, the authors did only conduct single-fault attacks and they did not attack a complete pairing computation. Nevertheless, their results confirm the threat that fault attacks pose to the security of pairings and their applications.

We will now present our results. They are based on [31]. We invert a pairing with the help of faults. We induce two faults in the computation of a pairing with two input points P and Q, the first of which is public. The goal of the attack is to reveal the second input point Q, which is secret. The faults facilitate the mathematical

cryptanalysis and the first argument pairing inversion problem (FAPI-1). In the literature, FAPI-1 is usually split into two parts: the exponentiation inversion and the Miller inversion, see Section 2.1.2. In the case of successful fault inductions, our first fault will facilitate the Miller inversion and we do not have to solve the exponentiation inversion problem, but skip the final exponentiation completely.

In Section 5.1, we start with the description of the low-cost setup that we used for the fault induction. In Section 5.2, we describe how this setup can be used to conduct single-fault and higher-order fault attacks against pairing computations. We explain how we utilized this setup to successfully conduct a practical second-order fault attack against a free software implementation of the eta pairing η_n on an ATXMega128A1 from the Atmel AVR family. This microcontroller was also one of the two devices that we attacked with the photonic side channel attack. We used this freely programmable chip to validate our attacks on a real-world smartcard platform. For our target device, the eta pairing was the only publicly available implementation with acceptable performance. Although the eta pairing is no longer recommended for security applications [7, 18, 81], the use of this pairing emphasizes the relevance of this fault attack since it was implemented by a third party. Implementations from the same crypto library have also been used in TinyPBC for the implementation of PBC in wireless sensor networks [127]. In Section 5.3, we resume the description of the second-order fault attack by explaining how the faulty pairing computation can be analyzed to reveal the secret input point. We present the cryptanalysis for three examples. Countermeasures to prevent such attacks are discussed in Section 5.4.

5.1 Experimental Setup

First, we describe in Section 5.1.1 the low-cost setup that we developed for the second-order fault attack. A block diagram of the setup is shown in Figure 5.1, and Figure 5.2 shows a picture of the system. It consists of three parts: the clock glitcher, the host system, and the target device. We used this system to generate instruction skip faults, i.e., transient faults which skip parts of the executed code by glitching the clock of the Central Processing Unit (CPU). Clock glitches are modifications of the frequency of the clock, i.e., modifications of the period of the clock cycle. Thus, clock glitching means to disturb the clock of the target device purposely so that the device operates different from its normal behavior due to the modified clock. Section 5.1.2 outlines how this setup can be used to perform instruction skip faults.

The development of the setup was mainly done by P. Günther and R. Gomes da Silva. A detailed description of the low-cost glitching platform and its functionality can be found in [80]. Our setup is similar to the setup of [15]. It is not specialized to attacks on pairings and can be used in other scenarios.

5.1.1 Low-Cost Glitching Platform

The glitcher is used to generate the external clock for the target device. It is also used to generate the glitches on the clock signal. The host system is used to configure

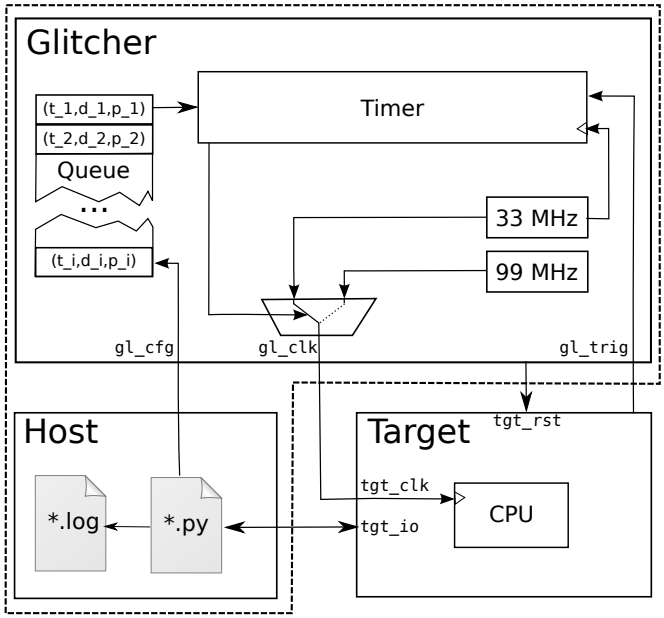


Figure 5.1: Block diagram of the setup for clock glitching. The host configures the glitcher, which generates the glitches on the external clock of the target device, and logs the output from the target device. The target device executes the attacked cryptographic pairing.

the glitcher and to acquire the output of the DUA, which executes the attacked program. We now describe the three components individually.

Clock Glitcher

For the hardware of the glitcher we use Die Datenkrake (DDK), an open source hardware peripheral for hardware analysis [4, 124]. It is a low-cost hardware and software toolchain for hardware security analysis, consisting of an ARM Cortex M3 processor [3] and an FPGA. The FPGA is used to perform the timing critical parts such as generation of the target's clock signal. The ARM CPU is mainly used to interface the FPGA with the host system. It implements a serial terminal that provides external control of the FPGA and an easy automation of the setup.

The glitcher has four IOs that are relevant for us: `gl_reset`, `gl_trig`, `gl_clk`, and `gl_cfg`, see Figure 5.1. The output `gl_clk` provides the clock signal for the DUA. The glitcher uses two internal clocks: a low frequency clock at $f_l = 33$ MHz and a high frequency clock at $f_h = 99$ MHz. The FPGA implements a 32-bit timer that manages the timing of different events. The clock source of the timer is f_l. The output `gl_clk` can be switched between f_l and f_h. The output `gl_reset` is connected to the target's external reset to enable an automated reset of the target. The input `gl_trig` is used for synchronization. Its main functionality is to reset the timer. Therefore, it is connected to `tgt_trig` in our setup. To fill a queue which is

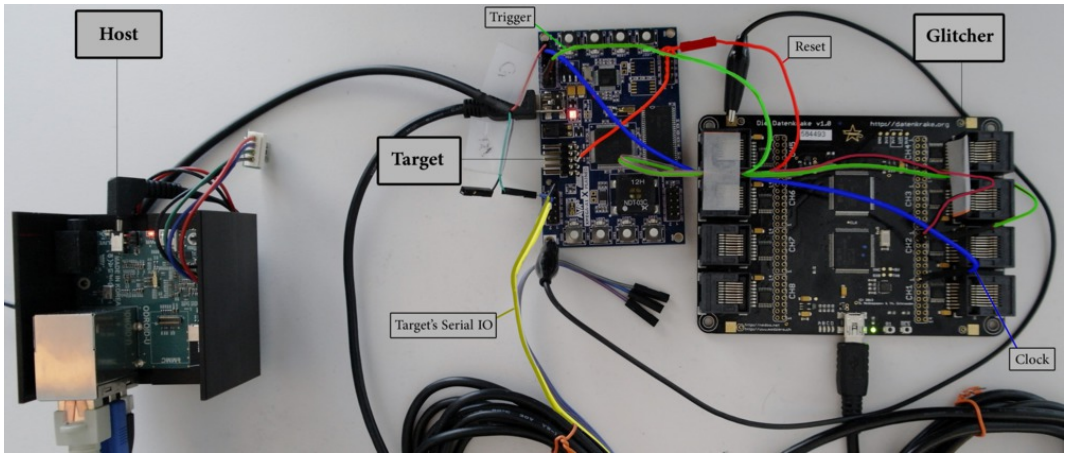


Figure 5.2: The DDK (glitcher), located on the right, provides the clock (blue) and reset signal (red) to the target, which is the ATXMega128A1 located in the center. The target also provides back to the DDK the trigger (green) indicating the beginning of the computation. The ODROID-U2 board (host), to which both the target's serial IO (yellow) and the DDK's console are connected to, can be seen on the left. The host configures and monitors the other devices.

necessary for a higher-order attack, the glitcher's internal ARM CPU listens to the serial input at `gl_cfg`.

A glitch is defined by three parameters: a timestamp t, a duration d, and a pattern p. When the timer reaches t, a glitch is generated by a synchronized switch from f_l to f_h for d periods of f_l, i.e., $3 \cdot d$ periods of f_h. We implemented two glitch patterns. For $p = 1$, the high frequency clock f_h is directly used to generate the glitch. For $p = 2$, the clock is gated during the second half of the f_l clock period, i.e., f_h is only used during the first half of the attacked clock period.

Since it is crucial for higher-order attacks to perform multiple synchronized glitches, the glitcher implements a glitch queue in First In - First Out (FIFO) order. This queue can be filled with up to 256 triples $(t_1, d_1, p_1), \ldots, (t_{256}, d_{256}, p_{256})$. Each triple consists of 48 bits, 32 of which are used for the time stamp, 8 are used for the delay, and 8 are used for the mode. Then, for every element in the queue, the corresponding glitch is generated. For second-order attacks, only two glitches need to be scheduled in the glitch queue.

Figure 5.3 shows two glitches. The first glitch is introduced with a delay of $t_1 = 3$ cycles of the 33 MHz clock, measured relatively to the trigger `gl_trig`. The glitch lasts for $d_1 = 2$ clock cycles. With $p_1 = 1$, the 99 MHz clock is directly used to generate the glitch pattern. The second glitch is introduced with a delay of $t_2 = 2$ cycles of the 33 MHz clock, i.e., it begins two clock cycles after the end of the first glitch. The second glitch lasts for $d_2 = 1$ clock cycle. With $p_2 = 2$, the 99 MHz clock is gated in the second half of the 33 MHz clock cycle. During a glitch, the delay between two consecutive positive clock edges is $\Delta = 1/f_h$.

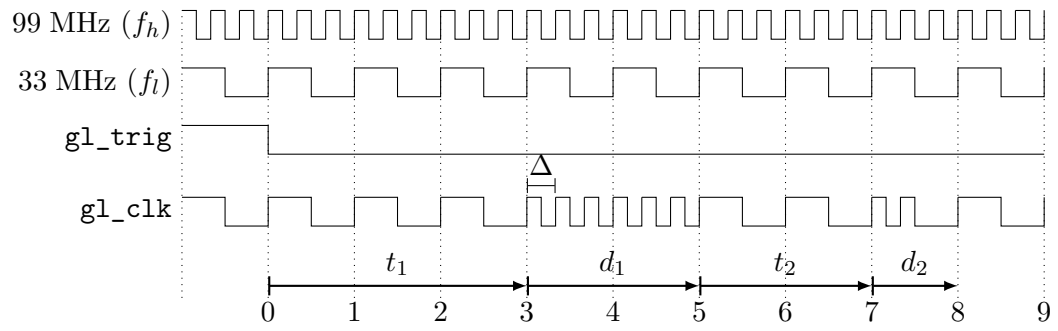


Figure 5.3: Two different glitches induced by the output `gl_clk` of the glitcher are shown. The first glitch is introduced with a delay of $t_1 = 3$ cycles of the 33 MHz clock, measured relatively to the trigger `gl_trig`. Its duration is $d_1 = 2$. With $p_1 = 1$, the 99 MHz clock is directly used to generate the glitch pattern. The second glitch is introduced with a delay of $t_2 = 2$ cycles of the 33 MHz clock, measured relatively to the end of the first glitch. Its duration is $d_2 = 1$. With $p_2 = 2$, the 99 MHz clock is gated in the second half of the 33 MHz clock cycle. During a glitch, the delay between two consecutive positive clock edges is $\Delta = 1/f_h$.

Host System

The host is a Linux-based system that configures the glitcher and automates the setup. We used an ODROID-U2 board [5]. However, other systems can also be used. The host provides two serial IO lines, `hst_io_1` and `hst_io_2`. The first one is used to configure the glitcher and is connected to `gl_cfg`. The second one is used to receive the output from the target and is connected to `tgt_io`, see Figure 5.1. The host system includes a Python library to interface with the glitcher [6]. This allows in-place analysis and logging of the target's output, followed by a direct reconfiguration for the next tests. These tests were required for our attack and involved checking many combinations of parameters (t_1, d_1, p_1) and (t_2, d_2, p_2) for the first and the second glitch, respectively. To generate those combinations, a Python script was used. For each experiment, the queue of the glitcher needs to be configured. This queue can hold up to 256 values, while we used only the values (t_1, d_1, p_1) and (t_2, d_2, p_2), corresponding to the two glitches. The output of the target device is written to a log file for further analysis. Another functionality provided by the host is to periodically execute a self-test routine for testing the integrity of the setup.

Target

We attacked an ATXMega128A1 from the Atmel AVR family [2]. In general, we assume that the target provides IO pins for at least reset input, clock input, trigger output, and serial IO. Analogous to the notation above, we denote these pins with `tgt_rst`, `tgt_clk`, `tgt_trig`, and `tgt_io`.

We control the CPU clock by connecting `tgt_clk` to the glitcher output `gl_clk`. Thus, the CPU of the target device is clocked with an external clock. This is not only given under laboratory conditions: low-cost implementations often rely on an external clock, either because no internal clock is available, or because it is very inaccurate and slow and even prohibiting stable serial communication. For an automated reset of the target, the glitcher controls the target's reset pin, see Figure 5.1.

For our concrete attack, we assume that the target generates a trigger on output `tgt_trig` before the computation of the target program is started. This signal is used to synchronize the target with the glitcher via `gl_trig`. Generating the trigger on the target is used to simplify the setup. In a real attack, it has to be generated by other means, e.g., it could be derived from sniffing the targets IO to locate the command that initiates the attacked computation. Then, the last edge on the serial line `tgt_io` could be used to externally generate a trigger. Finally, the serial IO `tgt_io` is used for the transfer of the executed cryptographic protocol. The IO `tgt_io` is connected to the host for initiating the attacked computation and for returning the result of the pairing.

5.1.2 Instruction Skips

The setup was used to produce precise clock glitches which lead to instruction skip faults. These instruction skips can be used to provoke various effects on the computation of any algorithm [140]. In Section 5.3, we will show several effects they can have on the computation of a cryptographic pairing. In case a `for` loop is attacked, for example, the number of iterations of the `for` loop can be modified by skipping a `jmp` or `branch` instruction. Clock glitches can not only generate instruction skip faults, but also instruction replacement faults and data corruption faults [15].

Introducing faults by clock glitching is done by systematically overclocking the DUA at defined instructions. Figure 5.3 shows a waveform of the target CPU clock and an example of two different clock glitches induced by the output `gl_clk` of the glitcher. During a glitch, the delay between two consecutive positive clock edges is $\Delta = 1/f_h$. If this difference Δ of two consecutive edges is outside of the functional range of the CPU circuit, there is a fair chance that the CPU computation gets disturbed. A potential disturbance is that the opcode of the current instruction is altered to a non-existing opcode. AVR CPUs ignore invalid opcodes during program execution [15]. Thus, altering an opcode to a non-existing opcode results in instruction skips.

5.2 Physical Attack

In this section, we describe the concrete physical attack that we performed against a free software pairing implementation. We used the setup explained in the previous section to perform a second-order fault attack. We skipped two instructions in the computation of the eta pairing. With the first fault we attacked a specific instruction

of the Miller Algorithm and with the second fault we completely skipped the final exponentiation.

In Section 5.2.1, we first explain how higher-order instruction skip attacks can be performed with the setup from Section 5.1. Such attacks are split into two phases, a profiling phase and a target phase. The profiling phase is used to learn relevant characteristics of the target implementation running on a test device. It is required only once. Then, in the target phase, the attack can be performed against any similar victim device which runs the same algorithm and stores arbitrary secrets.

In Section 5.2.2, we introduce the DUA and explain our concrete attack on the eta pairing. We explain the second-order attack along the two phases, profiling phase and target phase.

5.2.1 Realization of Higher-Order Fault Attacks

Although the system allows up to 256 faults, we explain the realization of higher-order attacks by presenting a second-order attack since we need only two faults and glitches, respectively, to successfully attack our target pairing.

To configure the glitcher from Section 5.1, the timing t, the duration d, and the pattern p of each glitch are required. The timing depends on the secret argument of the pairing. Hence, the timing is a priori unknown to us, which makes it challenging to determine t_1 and t_2. Thus, we execute a profiling phase to find reasonable configurations (t_1, d_1, p_1) and (t_2, d_2, p_2) for the two glitches. Once this profiling phase is completed, we do not need to repeat it when we attack new secrets on similar devices running the same pairing implementation. Thus, this profiling phase is required only once, even if an attacker wants to reveal secret inputs from several pairing computations.

Without loss of generality, we assume that the second argument $Q \in \mathbb{G}_2$ is the secret point. In a pairing-based cryptographic protocol, either the first or the second argument of the pairing may be secret, but the description of this attack is valid for both cases. Thus, assuming that the second argument $Q \in \mathbb{G}_2$ is secret allows us to give a more concrete description of the attack, but does not restrict its validity.

Profiling Phase

The profiling relies on two assumptions:

- The assembly code of the target pairing implementation is known to us.
- We are able to execute arbitrary profiling code on a profiling device which is similar to the target device.

Based on these assumptions, we first execute a modified pairing implementation on the profiling device. We modify the implementation in one or more of the following ways:

- We are able to compute the pairing for different values of Q that are chosen by us.

- We implement triggers T_1 and T_2 on two external IO pins. The trigger T_1 is raised immediately before the first target instruction and the second trigger T_2 is raised immediately before the second target instruction.
- We implement an emulation mode that branches over the first target instruction from the assembly. This emulates successfully skipping the first target instruction.

These modifications allow us to determine the timings t_1 and t_2, i.e., the clock cycles of the two target instructions, in every computation of the modified pairing. We do not measure t_2 from the start of the computation, but relative to t_1. Thus, we use the emulation mode to measure t_2, since we are interested in the delay in case the first fault has been successful. We execute the modified implementation for different secrets Q chosen uniformly at random from $\mathbb{G}_2$. As a result, we obtain distributions for t_1 and t_2. Since these distributions are obtained over the random choices of Q, we will choose the timing parameters in the target phase according to them.

These steps of the profiling can be done either by an oscilloscope or by programming a special profiling mode into the FPGA of the glitcher. The profiling mode counts the number of clock cycles between the start of the program execution `tgt_trig` and the trigger T_1, and between the triggers T_1 and T_2.

In the next step of the profiling, we determine useful combinations of the remaining glitching parameters d_1, d_2, p_1, and p_2. For that purpose, we perform a large number of experiments where we use the glitcher to introduce glitches close to the target instructions. For the parameters d_1, d_2, p_1, and p_2, many different values are tested. In this step, we use the fact that we know the values of Q in the profiling phase. Hence, we can predict the output of the algorithm when successfully glitching either one or both of the target instructions. This allows us to identify successful tests and their respective parameters.

Target Phase

In the subsequent target phase, the real attack takes place. The actual DUA with the unmodified code and the unknown secret is attacked. Since the glitching parameters depend on the secret, we do not know the successful combinations in advance. Therefore, we perform a sequence of experiments with different combinations of (t_1, d_1, p_1) and (t_2, d_2, p_2) until we are successful in skipping the two target instructions. We select the combinations and their order based on the results of the profiling phase.

5.2.2 Second-Order Fault Attack against the Eta Pairing

We practically conducted a second-order fault attack against the eta pairing $\eta_n(P, Q)$ on an ATXMega128A1 from the Atmel AVR family [2]. AVR controllers are also used in modern smartcards, while our version is freely programmable. A microcontroller from the AVR family was also analyzed in [15]. To attack an existing

pairing implementation which is commonly used, we used the RELIC toolkit [12]. The RELIC toolkit has also been used in TinyPBC for the implementation of PBC in WSNs [127]. It includes `C` implementations of finite field arithmetic, ECC, and PBC for different hardware platforms like Atmel's AVR family. To the best of our knowledge, this is the only freely available implementation of PBC for AVR CPUs. For our attack, we used RELIC version 0.3.5 without modifications of the source code. We compiled the library with the `avr-gcc-4.8.2` toolchain and optimization level `-O1`. We had to disable some low-level AVR-specific arithmetic optimizations (change `ARITH` from `avr-asm-271` to `easy`) since these optimizations resulted in incorrect pairing results for our target device, the AVR ATXMega128A1. However, this modification did not affect the source code itself, but only compile switches which deviate from the default settings of RELIC for AVR devices. The RELIC AVR default configuration defines the eta pairing (function `pb_map_etats()`) as the standard pairing. We chose the RELIC toolkit and the eta pairing to emphasize that we actually realized a fault attack against PBC. Thus, we chose an existing implementation that has already been used in other applications and did neither modify it nor build an artificial implementation with very easy attack vectors.

In our experiments, both arguments P and Q were loaded from the internal memory. Then $\eta_n(P, Q)$ was computed on the target device after which the output was returned on the serial IO `tgt_io`. Loading the public argument P from memory and not via the serial line helps to simplify the setup, but is not essential for the attack.

In our practical second-order fault attack against the eta pairing, we used the first fault to skip a specific instruction during the execution of Algorithm 2.1. We placed this fault at Line 9 of Algorithm 2.1 such that the jump to Line 2 was skipped. Hence, the `for` loop was terminated after the first iteration. To understand how we really attacked the end of the `for` loop, we refer to Table 5.1. It shows how the compiler generates the end of the `for` loop. An instruction skip fault that removes the `rjmp` instruction in Line 7 causes the loop to terminate immediately. We introduced the second fault at Line 10 of Algorithm 2.1 to skip the procedure call to the final exponentiation. A successful attack gave us a faulty computation where the `for` loop was executed exactly once and the final exponentiation was not executed at all. Hence, we only had to solve a facilitated Miller inversion afterwards. In Section 5.3, we will explain the cryptanalysis to obtain the secret argument of the pairing after a successful attack.

Profiling Phase

In the first step, we estimated t_1, the clock cycle of the `rjmp` instruction. Since we could choose the public point P both in the profiling phase and in the attack phase, we measured t_1 for a fixed value of P and different values of Q. We executed approximately $32,000$ experiments with random choices of $Q \in \mathbb{G}_2$ and measured

```
1  call fb4_mul_dxs
2  .LVL43:
3  subi r16, 1      ; decrement loop counter
4  sbc r17, __zero_reg__
5  .loc 1 247 0 discriminator 2
6  breq .+2
7  rjmp .L2         ; jump to loop begin
8  .LBE2:
9  .loc 1 486 0
10 subi r28, 36     ; clean stack
11 sbci r29, -2
12 out __SP_L__, r28
13 out __SP_H__, r29
14 pop r29
```

Table 5.1: Assembly code of the end of the `for` loop, generated with `avr-gcc`. We used the first fault of our second-order fault attack to skip the instruction of Line 7 to terminate the loop after its first iteration.

t_1 for each experiment. The value of t_1 is not constant, since several computations depending on the secret input point are performed during the first iteration of the loop. The distribution of t_1 is given in Table 5.2.

Then, we determined t_2. We recall that t_2 is not the number of clock cycles from the start of the computation to the call of the final exponentiation. It is measured relatively to t_1, i.e., it is the number of clock cycles from the `rjmp` to the call of the final exponentiation. We used the emulation mode of the profiling code which allows us to skip the `rjmp` instruction at t_1. We obtained a constant value for t_2 since in case the first glitch is successful in leaving the `for` loop, the code executed between t_1 and t_2 is data-independent and therefore constant. We obtained the value $t_2 = 28$.

After having learned the distribution of t_1 and the value of t_2, we injected approximately $40,000$ faults to select promising combinations of the glitch durations d_1 and d_2 and the glitch patterns p_1 and p_2 for the target phase. For the duration of the glitches, we found $d_1 \in \{3, 5\}$ and $d_2 \leq 5$ as reasonable settings for the target phase. Regarding the two patterns p_1 and p_2 depicted in Figure 5.3, both produced good results, but we did not detect critical differences in their effects. Hence, we used both patterns in the target phase in order to ensure a successful attack. We needed less than 72 hours for this selection process. Since we knew the secret input Q and thereby also the values of t_1 and t_2 during the profiling phase, we were always successful in introducing the faults at the correct instructions.

t_1 in instruction cycles	occurrence	in %
422,780	1	< 0.01
424,515	1	< 0.01
424,941	1	< 0.01
427,731	1	< 0.01
431,069	1	< 0.01
581,804	3	0.01
581,903	28	0.08
582,001	7	0.02
582,002	590	1.66
582,100	30	0.08
582,101	1,763	4.95
582,111	1	< 0.01
582,199	297	0.83
582,200	32,890	92.35

Table 5.2: Distribution of t_1, the timing of the first instruction skip. This is the execution time of the `rjmp` instruction in Table 5.1. The execution time was measured based on a fixed public point and different secret inputs. It is not constant since it depends on the secret input.

Target Phase

Based on our results from the profiling shown in Table 5.2, we scheduled t_1 as $582,200 - i \cdot 99$ for $i \in \{0, \ldots, 5\}$. We did not observe results for $t_1 = 582,199$ since we blame these occurrences as inaccuracy and hence count them as delay 582,200. If we did not succeed with one of these values, we fell back to a brute force search with $t_1 = 582,200 - i$ for $i = 1, 2, 3, \ldots$ until we were successful. We combined all potential values of t_1 with all potential combinations of d_1, d_2, p_1, p_2, and t_2 that we determined in the profiling phase. For t_2 we added a small safety margin such that $t_2 \in \{26, \ldots, 30\}$. Furthermore, we repeated each combination for ten times since even with the correct parameters, glitching is not always successful. Thus, for each value of t_1 we performed $2 \cdot 5 \cdot 2 \cdot 2 \cdot (30 - 25) \cdot 10 = 2,000$ experiments. For our setup, one test required 7.5 seconds on average. This includes configuration of the glitcher, communication from target to host, and self-tests. Hence, we were able to perform more than 10,000 experiments per day.

We repeated the attack for five different secrets, drawn uniformly at random from $\mathbb{G}_2$. We were always successful in skipping both instructions. The analysis of these experiments shows that t_1 was either $582,200$ or $582,101$ for all five secrets. This is in accord with the distribution in Table 5.2. Hence, for each attack we required at most $2 \cdot 2,000$ experiments, whereas in the cases with $t_1 = 582,200$, much fewer experiments were required and it took us only minutes to be successful.

We will show in Section 5.3 how the results of such successfully corrupted computations can be analyzed to reveal the secret input Q of the pairing. Furthermore,

we will show that we are even able to efficiently determine from the target's output whether or not both target instructions were skipped successfully. Hence, the results can be analyzed on the fly during the target phase and all remaining experiments can be discarded once the first successful experiment is detected.

Note that a short execution time is not sufficient to determine that both glitches were successful. While two successful glitches always lead to a short execution time, a short execution time can also be caused by glitches that lead to faults other than the intended ones.

5.3 Cryptanalysis

We resume the description of the second-order fault attack by explaining the mathematical analysis which leads from the faulty computation to the secret input. In Section 5.3.1, we describe the analysis for our concrete attack against the eta pairing, i.e., the attack from Section 5.2. Next, we describe two more analyses in Sections 5.3.2 and 5.3.3, both of which also assume a second-order fault attack where the first fault targets the computation of the Miller Algorithm, while the second fault is used to skip the final exponentiation completely. First, we explain another attack vector against the eta pairing computation. Thus, this attack can be conducted against the ATXMega128A1 with the setup from Section 5.2, without any modification of the pairing implementation. Then, we describe the cryptanalytic details for the case that another pairing, the reduced Tate pairing, is attacked with a second-order attack. We analyze the same implementation of the Tate pairing as used in TinyTate, which is an implementation of the Tate pairing for sensor nodes [128].

Before we start with the description of these three cryptanalyses, we introduce a two-part categorization that covers all fault attacks against the Miller Algorithm from an analytical perspective. We introduce this categorization since from the analytical perspective, the physical realization of the fault attack is not relevant. Moreover, different physical faults or fault injection techniques such as clock glitches, power glitches, and laser fault attacks, may lead to the same effect on the algorithm.

Some works contain already categorizations of fault attacks. However, these are not unambiguous and therefore not helpful for a precise description. Fault attacks have been categorized as having three main effects on an algorithm: knock out a step in the computation, cause a loop to either end prematurely or run over, and to cause the data being operated on to be corrupted in some way [180]. In the same work, the authors also considered the locations that a data corruption fault can target in the Miller loop. Regarding fault attacks on pairing computations, faults were also described as corrupting precomputed values or parameters, inputs to the pairing, and intermediate values [65]. These criteria are helpful to describe fault attacks on a high level, but they are not unambiguous: a fault which knocks out a step in the computation so that the loop runs over cannot be uniquely categorized in accordance with [180]. A fault in a program flow which alters the public input

P after some iterations of the loop and thus, also alters the intermediate values, cannot be uniquely categorized in accordance with [65].

Relevant to the effects a single fault can have on the Miller Algorithm, there are only two distinct categories in our opinion: a fault can either be modeled as having modified the Miller bound n, or it can be modeled as having modified the Miller variable f.

Modification of n: In this category we classify all faults that can be modeled by a modification of the Miller bound n to n' [13, 63, 65, 132, 133]. This includes the following interesting attacks:

- Modification of n to n' while loading the loop counter.
- Modification of n to n' directly in memory [63].
- Early termination of the Miller loop.
- Skipping of conditional if branches [13].
- Corruption of pointer to the Miller variable.

Modification of f: This category includes all faults which result in a modification of the Miller variable f [65, 180]. The Miller variable is updated during all iterations of the Miller loop. Thus, it can be modified during any iteration of the loop. Note that the actual fault does not have to alter f directly, but, e.g., the intermediate point T, Algorithm 2.1. However, this will result in a modified computation of f. This category includes the following interesting attacks:

- Disturb loading of P or Q during line computations.
- Skip update of point during line computations.
- Corrupt a field element directly in memory [180].
- Sign change fault attack [180].

With our setup from Section 5.1, we can realize any fault from both categories and thereby all theoretical faults against the Miller Algorithm that have been presented so far.

The practical example from Section 5.2 and thus, the corresponding analysis from Section 5.3.1, falls into the first category. The other two examples from Sections 5.3.2 and 5.3.3 fall into the second category, so that both categories are covered by the three cryptanalytic examples in this section. Similar examples are known, e.g., [63, 65, 133, 180]. In all three examples, we assume to know the first pairing input $P = (x_P, y_P)$, while $Q = (x_Q, y_Q)$ is secret. For the description of the analyses, we define variables x and y representing the x-coordinate x_Q and the y-coordinate y_Q of the secret Q. Moreover, we assume that the first fault is induced during the

computation of the Miller Algorithm, while the second fault skips the function call to the final exponentiation. Thus, the exponentiation inversion does not have to be solved at all, but only a facilitated Miller inversion.

5.3.1 Modification of n in the Eta Pairing

Before we explain how to analyze the output of our second-order attack from Section 5.2, we provide mathematical details of the attacked implementation. For the analysis, we need the output of the attacked computation and the output of one error-free computation of $\eta_n(P, Q)$.

We attacked an implementation of the eta pairing in characteristic 2 on supersingular elliptic curves. We decided to attack the eta pairing despite current research results which indicate that it should no longer be used for security applications [7, 81]. This was due to the fact that the eta pairing is the default for AVR devices in the attacked RELIC library [12]. However, the attack can be easily applied to other pairings as we will show in Section 5.3.3.

In our attacked implementation, the elliptic curve $E : y^2 + y = x^3 + x$ is defined over the finite field $\mathbb{F}_q$ with $q = 2^m$ and $m = 271$. For our case, i.e., $m = 7 \mod 8$, it holds that $\#E(\mathbb{F}_q) = 2^m + 2^{(m+1)/2} + 1$. We define the extension field $\mathbb{F}_{q^4} = \mathbb{F}_q(s, t)$ of degree 4 with $s^2 = s + 1$ and $t^2 = t + s$ and we define the distortion map ψ with $\psi(x, y) = (x + s^2, y + sx + t)$. Let

$$z = (q^4 - 1)/\#E(\mathbb{F}_q) = (2^{2m} - 1) \cdot (2^m - 2^{(m+1)/2} + 1) \tag{5.1}$$

and $n = 2^{(m+1)/2} + 1$. Furthermore, we have to replace P by $-P$ in the improved version of the eta pairing η_n as proposed in [19, Sections 4 and 6]. For input $P, Q \in E(\mathbb{F}_q)$, the eta pairing η_n is then defined as

$$\eta_n(P, Q) = f_{n,-P}(\psi(Q))^z. \tag{5.2}$$

Note that the eta pairing η_n is defined as a symmetric pairing in Equation 5.2 and that the domain of η_n in Equation 5.2 is different from the domain in Definition 2.21. The distortion map, however, maps into the second Eigenspace of Frobenius, as used in Definition 2.21. Due to the simple binary form of $n = 2^{(m+1)/2} + 1$, the main loop of Algorithm 2.1 reduces to point doublings and squarings of field elements in $\mathbb{F}_{q^4}$, followed by one multiplication with $l_{\left[2^{(m+1)/2}\right](-P),-P}(\psi(Q))$. This final multiplication comes from the least significant bit of n. As in [19, Algorithm 3], the eta implementation computes the loop in reversed order in the RELIC library [12]. Therefore, $P' = \left[2^{(m-1)/2}\right](-P)$ needs to be defined. Furthermore, the first loop is unrolled so that we yield

$$f_{n,-P}(\psi(Q)) = l_{[2]P',-P}(\psi(Q)) \cdot g_{P'}(\psi(Q)) \cdot \prod_{j=1}^{(m-1)/2} g_{[2^{-j}]P'}(\psi(Q))^{2^j}. \tag{5.3}$$

Algorithm 5.1 Implementation of $\eta_n(P,Q)$ on $E(\mathbb{F}_{2^m})$ for $m = 7 \mod 8$, $n = 2^{(m+1)/2}+1$, and $E : y^2 + y = x^3 + x$, as used in our practical fault attack.

Require: $P = (x_P, y_P), Q = (x_Q, y_Q) \in E$
Ensure: $\eta_n(P,Q)$

1: $u \leftarrow x_P$, $v \leftarrow x_Q$
2: $g \leftarrow u \cdot v + y_P + y_Q + 1 + (u + x_Q)s + t$
3: $u \leftarrow x_P^2$
4: $l \leftarrow g + v + u + s$
5: $f \leftarrow g \cdot l$
6: **for** $i = 1$ .. $(m-1)/2$ **do**
7: $\quad x_Q \leftarrow {x_Q}^2$, $y_Q \leftarrow {y_Q}^2$
8: $\quad x_P \leftarrow \sqrt{x_P}$, $y_P \leftarrow \sqrt{y_P}$
9: $\quad u \leftarrow x_P, v \leftarrow x_Q$
10: $\quad g \leftarrow u \cdot v + y_P + y_Q + 1 + (u + x_Q)s + t$
11: $\quad f \leftarrow f \cdot g$
12: **end for**
13: $f \leftarrow f^z$
14: **return** f

Algorithm 5.1 shows how the computation of (5.3) is implemented in the RELIC library for our specific case $m = 271 = 7 \mod 8$ and $b = 0$. The concrete implementation is very similar to the implementation proposed in [19, Section 6].

For this RELIC implementation, the two instruction skip faults target the first execution of Line 12 and the execution of Line 13 of Algorithm 5.1. Table 5.1 shows the generated assembly for Line 12 of Algorithm 5.1. The first fault is used to skip the `rjmp` instruction in Line 7 and causes the loop to terminate immediately. In a fault atack where both fault injections are successful, the `for` loop is executed exactly once and the final exponentiation is completely skipped. However, since one loop iteration is unrolled in this implementation of the eta pairing, this corresponds to an execution with two iterations of the respective loop in Algorithm 2.1, and a modification of n from $2^{(m+1)/2}+1$ to 2^2+1. Thus, this attack belongs to the category of faults which modify n.

We denote the faulty output of the attacked computation with α. Hence, for a successful injection of the two faults it holds that $\alpha = f_{n',-P}(\psi(Q))$ with $n' = 2^2+1$. With (5.3) we obtain

$$\begin{aligned} \alpha &= f_{n',-P}(\psi(Q)) \\ &= l_{[2]P',-P}(\psi(Q)) \cdot g_{P'}(\psi(Q)) \cdot g_{[2^{-1}]P'}(\psi(Q))^2. \end{aligned} \tag{5.4}$$

The following steps describe how we recover the secret input Q of the pairing from α.

1. **Algebraic Model of the Secret:** We describe Q as the root of a rational function f_P. With Equation 5.4 we define

$$f_P(x_Q, y_Q) := f_{n',-P}(\psi(x_Q, y_Q)) - \alpha. \tag{5.5}$$

   Since $f_{n',P}(\psi(x_Q, y_Q))$ is a product of four lines, $f_P(x_Q, y_Q)$ is of degree at most 4 in x and y. The secret $Q \in E$ is defined over the subfield $\mathbb{F}_q$ of $\mathbb{F}_{q^4}$. Therefore, we consider $\mathbb{F}_{q^4}$ as a 4-dimensional vector space over $\mathbb{F}_q$. Then (5.5) can be re-written as four individual polynomials $f_P^{(1)}, \ldots, f_P^{(4)}$ over $\mathbb{F}_q$. This reduces the computational complexity of the analysis in the next step.

2. **Computation of Candidates:**

   In this step, simultaneous roots of $f_P(x_Q, y_Q)$ and E are computed. We define the algebraic variety $V_Q = V\left(f_P^{(1)}, \ldots, f_P^{(4)}\right) \cap E$ by a system of nonlinear multivariate equations. This system might also be overdetermined. Since $Q \in V_Q$, all elements of V_Q have to be computed. This can be done with Gröbner basis techniques [57]. Gröbner basis techniques are implemented in several computer algebra systems such as Magma [39] and Sage [163].

3. **Testing Candidates:** In the final step, we identify the secret from all elements in V_Q. To do this, we compute $\eta_n(P, Q')$ for each $Q' \in V_Q$. Each result is compared with $\eta_n(P, Q)$, which has been obtained from an error-free execution of the target pairing to identify the unique point Q.

   Alternatively, one can start with the calculation of the order of all candidates. We know the order of the correct point Q. Therefore, we can eliminate all candidates which are not of the respective prime order.

The case where P is the secret can be handled analogously. The main difference is that we replace $f_P(x, y)$ from Step 1 by a polynomial where x and y represent $\sqrt{x_P}$ and $\sqrt{y_P}$. From Algorithm 5.1 we see that the degree of $f_P(x, y)$ will now become at most $d = 7$.

Restricting to subfields as in Step 1 can often be exploited. It has already been used in [175] and [180]. This relies on a common optimization for the implementation of pairings, which is to choose the first argument P in $\mathbb{G}_1 \subseteq E(\mathbb{F}_q)$. Consequently, for Type 1 pairings the second argument Q is also $\mathbb{F}_q$-rational. For Type 3 pairings, Q is defined in $\mathbb{G}_2 \subseteq E'(\mathbb{F}_{q^{k'}})$, where E' is a degree k' twist of E and k' divides k.

As explained in Section 5.2, many experiments fail in skipping both target instructions in a single execution of the algorithm. In case the attacker did not succeed in skipping both target instructions as intended, no candidate Q' will satisfy the equation $\eta_n(P, Q') = \eta_n(P, Q)$ from Step 3, since for the fixed point P, $\eta_n(P, \cdot)$ is injective. Hence, during the analysis it is automatically detected whether or not the attack was successful. Therefore it is reasonable to automate the analysis on input α in a practical attack for identifying the first successful experiment, and

thereby the secret input, on the fly. Those experiments with a long runtime can be ignored in the analysis since here the attacker knows that the glitches did not work as intended.

We automated the analysis based on Sage [163], a free computer algebra system. We re-implemented the eta pairing from the RELIC library in Sage. Based on this implementation, we are able to automatically construct the multivariate polynomial (5.5) from Step 1 for any value α. From this multivariate polynomial, we derive $f_P^{(1)}, \ldots, f_P^{(4)}$. Step 2 is an invocation of the `variety()` function on the ideal generated by $f_P^{(1)}, \ldots, f_P^{(4)}$ and $y^2 + y = x^3 + x$. This computation is based on Gröbner bases. Finally, in Step 3 we use the implementation of the pairing to obtain the value $\eta_n(P, Q')$ for all the candidate points. These are subsequently used for comparison with the output of the error-free computation and the identification of the correct Q. Our non-optimized implementation requires less than one second for processing a single faulty output α. This is less time than the DUA requires to compute the pairing. Hence, the time for cryptanalysis is not critical for the performance of our attack.

5.3.2 Modification of f in the Eta Pairing

For this example, we assume that the same implementation of the eta pairing as in our practical attack is computed. We attack two computations of $\eta_n(P, Q)$. In both computations, the same input has to be used and in both computations, we skip the final exponentiation completely. During the first computation, this is the only fault that we induce. We denote the output with α_1, i.e., $\alpha_1 = f_{n,-P}(\psi(Q))$. In the second computation, we induce another fault during the computation of the Miller function before we skip the final exponentiation. This first fault consists in skipping an instruction which is involved in the update of the Miller variable f. Thus, this attack belongs to the category of faults which modify f. In the general description of the Miller Algorithm, this corresponds to the Lines 3, 4, 6 or 7 of Algorithm 2.1. In our concrete implementation, i.e., Algorithm 5.1, again several instructions can be skipped to achieve a modification of f. For this example, we choose to illustrate the modification of f by skipping the update of u once. Thus, either Line 3 or Line 9 in any round of the `for` loop in Algorithm 5.1 can be skipped. We choose to skip Line 3. We denote the second faulty output with α_2, i.e., $\alpha_2 = f'_{n,-P}(\psi(Q))$. Since α_1 and α_2 are known, we also know $\alpha = \alpha_1/\alpha_2 \in \mathbb{F}_{q^4}$. In addition to these two attacked computations, we also need the correct result of a third computation. This computation has to use the same inputs and is not attacked, but error-free.

1. **Algebraic Model of the Secret:** The two values α_1 and α_2 have the same first factor g, which is computed in Line 2, but differ in their factor l, which depends on u. The updated value of u in the next iterations of the Miller loop depends only on x_P, but not on u itself. Since x_P is not attacked in this scenario, all further factors of α_1 and α_2 which are computed during the `for`

loop are equal. Thus, since all but the respective factors l of α_1 and α_2 are equal, we receive the equation

$$\begin{aligned} &x_P \cdot x_Q + y_P + y_Q + 1 + (x_P + x_Q) \cdot s + t + x_Q + x_P^2 + s \\ &= \alpha \cdot [x_P \cdot x_Q + y_P + y_Q + 1 + (x_P + x_Q) \cdot s + t + x_Q + x_P + s]. \end{aligned} \tag{5.6}$$

All values except x_Q and y_Q are known. Thus, this algebraic model gives us one equation in the two unknowns x and y. The elliptic curve is defined by $E : y^2 + y = x^3 + x$ and gives us a second equation with root Q.

2. **Computation of Candidates:**

   We have two equations with two unknowns x and y. By writing both E and Equation (5.6) as univariate polynomials in y and using the theory of resultants, we get a univariate polynomial in x which has degree at most 3.

$$\begin{aligned} &\operatorname{Res}\left(\alpha \cdot f'_{n,-P}(\psi(x,y)) - f_{n,-P}(\psi(x,y)), E\right) \\ &= (\alpha - 1)^2 \cdot (-x^3 - x) \\ &\quad + \Big[(\alpha - 1)\Big(x_P \cdot x + x_P + x + y_P + 1 \\ &\quad + (x_P + x + 1) \cdot s + t\Big) - x_P^2 + x_P\Big]^2 \\ &\quad - \Big[(\alpha - 1)\Big(x_P \cdot x + x_P + x + y_P + 1 \\ &\quad + (x_P + x + 1) \cdot s + t\Big) - x_P^2 + x_P\Big] \cdot (\alpha - 1) \end{aligned} \tag{5.7}$$

   All roots of this polynomial are candidates for x_Q. For each of these candidates we evaluate Equation (5.6) and thereby get a candidate for the secret point Q.

3. **Testing Candidates:** Since we know the concrete implementation, we now compute $\eta_n(P, Q')$ for all candidates Q' and compare the results with $\eta_n(P, Q)$, which has been obtained from an error-free execution. Again, only the correct candidate will satisfy the equation $\eta_n(P, Q') = \eta_n(P, Q)$, since for the fixed point P, $\eta_n(P, \cdot)$ is injective. Hence, during the analysis it is automatically detected whether or not the attack was successful. Since Equation 5.7 has degree at most three, we have to test at most three candidates to identify the unique point Q.

   Again, one can start with the calculation of the order of all candidates since the order of the correct point Q is known. Therefore, all candidates which are not of the respective order can be eliminated.

The roles of x_Q and y_Q can be switched. However, the resulting univariate polynomial in y has at most degree four. Due to the higher degree, the analysis will

Algorithm 5.2 BKLS Algorithm for the computation of the reduced Tate pairing.

Require: $P \in E(\mathbb{F}_{q^2})[r], Q \in E(\mathbb{F}_{q^2})[r]$
Ensure: $t(P,Q)$

1: $f \leftarrow 1$
2: $A \leftarrow P$
3: $n \leftarrow r-1$
4: **for** $i = \lfloor \log(r) \rfloor - 2 \ldots 0$ **do**
5: $\quad f = f^2 \cdot g(A, A, Q)$
6: $\quad$ **if** $n_i = 1$ **then**
7: $\quad\quad f = f \cdot g(A, P, Q)$
8: $\quad$ **end if**
9: **end for**
10: $f = \bar{f}/f$
11: **return** $f^{(q+1)/r}$

become more expensive. With Equation (5.6), we will again get one candidate for the secret point for each root. Thus, we have to test at most four candidates.

As in the example of our concrete attack, the case where P is the secret can be handled analogously. The main difference is that Equation 5.6 is quadratic in x_P, while it is linear in both coordinates of Q. In the case where P is the secret, all values except x_P and y_P are known in Equation 5.6.

5.3.3 Modification of f in the Reduced Tate Pairing

This example targets the reduced Tate pairing. We attack two computations of $t(P,Q)$ in the manner of the previous example. In both computations, the same input has to be used and in both computations, we skip the final exponentiation completely. During the first computation, this is the only fault that we induce. We denote the output with α_1. In the second computation, we induce another fault before we skip the final exponentiation. This first fault consists in skipping an instruction which is involved in the update of the Miller variable f. Thus, this attack belongs to the category of faults which modify f. We denote the output of the second faulty computation with α_2.

Algorithm 5.2 shows how the computation of the reduced Tate pairing, i.e., Equation 2.16, is proposed for affine coordinates [20, 152]. This algorithm is named after its developers Barreto, Kim, Lynn, and Scott and is called BKLS algorithm. The same algorithm for projective coordinates is used in TinyTate, an implementation of the Tate pairing for sensor nodes [128]. For this example, we make use of the exemplary nonsingular curve $E : y^2 = x^3 - 3x + B$ with $B \in \mathbb{F}_q$, $q = 3 \mod 4$ and embedding degree $k = 2$ [152].

Now, we explain the representation of points on E [152]. Since we assume $q = 3 \mod 4$, every element from $\mathbb{F}_{q^2}$, i.e., each coordinate of an elliptic curve point, can

be regarded as a complex number $a + bi$ with $a, b \in \mathbb{F}_q$. Thus, any point $(x, y) \in E$ can also be represented with $x = [a, b]$ and $y = [c, d]$ for $a, b, c, d \in \mathbb{F}_q$. We assume that the public input is on $E(\mathbb{F}_q)$, i.e., the imaginary parts of these coordinates are zero. Thus, $P = (x_P, y_P)$ with $x_P, y_P \in \mathbb{F}_q$. Moreover, according to [152], we assume that the secret input Q is of the form $Q = ([a, 0], [0, d])$ with $a, d \in \mathbb{F}_q$. Thus, $Q = (x_Q, iy_Q)$ with $x_Q, y_Q \in \mathbb{F}_q$. We denote the complex conjugate of the Miller variable f with $\bar{f}$.

For inputs P, P, and Q, the function g used in the Miller loop is defined as follows:

$$g(P, P, Q) = y_P - \lambda(x_Q + x_P) - i \cdot y_Q \tag{5.8}$$

This is exactly the computation of the function during the first invocation of Line 5. Note that i does not denote the loop counter here, but the imaginary unit. The factor $\lambda \in \mathbb{F}_q$ denotes the slope of the tangent at P.

For this example, we choose to illustrate the modification of f by skipping the multiplication with $g(A, A, Q)$ once. To have a result which is easy to analyze, we explain the analysis in the case that this multiplication is skipped during its last invocation, i.e., for $i = 0$. Here, we have $A = [(r - 1)/2] \cdot P$. Thus, we skip $g([(r-1)/2]P, [(r-1)/2]P, Q)$. According to [152], the multiplication with $g(A, P, Q)$ is not processed for the least significant bit of r. Thus, by skipping $g(A, A, Q)$ during its last invocation, the final step of the Miller Algorithm is skipped. We denote the second faulty output with α_2, i.e., α_2 is the result of the computation when $g(A, A, Q)$ was skipped for $i = 0$ and the final exponentiation was skipped afterwards. Note that since $k = 2$, the exponent of the final exponentiation is $(q^2 - 1)/r = (q - 1)(q + 1)/r$. Only the exponentiation with $(q + 1)/r$ is actually included in the final exponentiation in Line 11, while the exponentiation with $(q-1)$ is computed in Line 10. This line, however, is not skipped in this example.

1. **Algebraic Model of the Secret:** We know the results $\alpha_1, \alpha_2 \in \mathbb{F}_{q^2}$. Therefore, we can compute the value $\alpha = \alpha_1/\alpha_2 \in \mathbb{F}_{q^2}$. With $A := [(r - 1)/2]P$ and due to the precise instruction skip, we can now isolate the factor $g(A, A, Q)$. We yield the function $f_P(x_Q, iy_Q)$ with root Q:

$$f_P(x_Q, iy_Q) := (\alpha - 1)\lambda x_Q + (\alpha + 1)iy_Q + (1 - \alpha)y_A + (\alpha - 1)\lambda x_A \tag{5.9}$$

   All values except x_Q and iy_Q are known.

2. **Computation of Candidates:**

   As in our first example, simultaneous roots of $f_P(x_Q, iy_Q)$ and E have to be computed. Again, this can be done with Gröbner basis techniques, and it can be automated using a computer algebra system. Since $f_P(x_Q, iy_Q)$ is linear both in x_Q and iy_Q, the complexity of this step is small.

3. **Testing Candidates:** Now, we test all candidates that were computed in the previous step. If we assume that Q has to be of prime order r, we can again start with the computation of the order of the candidates. Otherwise, we compute $t(P, Q')$ for each candidate Q' and compare it with a correct result of the computation $t(P, Q)$.

It might seem to be more difficult to skip an instruction at the end of a loop as it is at the beginning. However, our experiments for the timing of the first instruction skip in Section 5.2 show that the variation due to different values of Q is less strong than expected. Thus, this third exemplary attack is also realizable with only little additional effort.

Aside from that, the cryptanalysis of the faulty results would not be as easy as in this third example if the first invocation of that function would be skipped.

5.4 Countermeasures

The defense against this attack involves both the software and the hardware.

At the hardware level, the countermeasure has to detect attempts of glitching. This can be ensured by suitable sensors which detect glitches [140]. Another hardware-level defense against the clock glitches that we used is that devices which are interesting targets for such an attack work only for certain specified clocks but refuse to work at higher clocks.

At the software level, generic countermeasures like checksums and redundant computations might prevent fault attacks. However, they might be too expensive in terms of space or time or not effective against all types of faults in the pairing-based context, as this turns out to be more complex than traditional cryptography. Hence, mitigations tailored to efficient pairing computations on resource-constrained devices have to be developed.

Regarding fault attacks against the Miller loop, it has to be ensured that the whole loop is actually computed. This, however, is not as simple as it might seem. It was suggested to protect the Miller loop by ensuring that the correct number of iterations has been computed, e.g., by verifying the counter or monitoring the timing [110]. However, this is not sufficient. Assuming that the pointer to the intermediate result is corrupted by a fault attack, the intermediate result is not updated anymore and the attacker obtains the same value as if she had forced the Miller loop to end prematurely. Thus, ensuring that the correct number of iterations has been computed is only one contribution to securing the Miller loop. The second requirement is that the result of the complete loop computation is actually further processed afterwards.

A simple but expensive method of ensuring that the whole loop is actually computed is to duplicate the computation and to store the results in different registers before comparing them. With an additional fault however, this can also be circumvented by an experienced attacker, either by inducing the same fault in both

computations or by skipping the check. Therefore, computing the Miller loop twice is neither sufficient as isolated countermeasure.

As can be seen in our examples, the mathematical complexity of the subsequent analysis is particularly feasible when the faults are induced at the very beginning or the very end of the Miller loop. The algebraic model of the secret has only a small degree in x_Q and y_Q in these situations and is therefore easily solvable. Hence, it is particularly important to secure the operations of these outer iterations. The importance of the protection of the first and last rounds is also known from block cipher cryptanalysis [10, 47].

The precise timing of the instruction skips is of upmost importance. Thus, random delays and randomized dummy operations before and during the execution of the Miller loop will complicate such attacks considerably since it will be more difficult to find the correct position [47].

Randomization of group elements relying on redundant representations, such as randomization of Jacobian coordinates, is not effective against fault attacks [110]. This is obvious since such randomization techniques exploit redundant representations of the group elements, but do not process different group elements. Hence, these techniques lead to variations of the side channel leakage without modifying the values of the computation. Therefore, they might only help against differential side channel attacks. Checking whether or not the temporary points lie on the correct curve neither helps, since they always do in our examples. Checking whether or not the final result of the pairing computation is in μ_r, by contrast, is an effective countermeasure. However, since checks can be skipped with an additional fault, this has to be carefully implemented.

Blinding techniques which lead to the computation of a pairing with different input points, however, are a good way to protect pairing computations against fault attacks [133]. Page and Vercauteren propose additive and multiplicative blinding techniques. These lead to the computation of the same pairing for different input points and can be reversed with a multiplication and - depending on the blinding - an additional pairing computation. By blinding the public input point, i.e., the point under control of the attacker, this control is effectively removed. By blinding the secret input point, the attacker can only learn the blinded point. In the case that such blinding techniques are used, however, the random points which are added and the random factors, respectively, must not remain constant during the lifetime of the cryptographic device, but have to be changed often. Otherwise, the attacker can learn them and hence circumvent this countermeasure.

Our successful attack highlights the demand for further research on how to protect against the complete skipping of the final exponentiation. In our implementation, the complete skipping was possible since we compiled the RELIC library with optimization level `-O1`. This attack vector does not exist if one compiles the library with optimization level `-O2` instead. Then, the compiler replaces the function call to the final exponentiation with inline code, i.e., the call is replaced by many lines of code in the body of the program. Thus, our approach of completely skipping the final exponentiation with a single instruction skip is blocked in this scenario. In

other realistic implementations, the complete skipping might also be impossible due to code optimizations. Hence, code optimizations can prevent the complete skipping of the final exponentiation. This might prevent the attack on the final exponentiation or at least complicate it considerably, since in the case of an attack against the optimized code, an improved mathematical analysis is necessary.

It is worth noting that for most cryptographic protocols that are based on pairings, fault attacks do not pose a serious threat. This is because the result of the pairing usually cannot be directly accessed by the attacker since, e.g., a cryptographic hash function is applied to the pairing result directly [38]. The skipping of the hash function is not as easy as it might seem, since the domain of the pairing and the domain of the hash function are usually different. Thus, skipping the hash function might lead to unintended complications. We see that the direct application of a hash function to the result of a pairing prevents the release of this result in the case that the hash function is carefully implemented. Therefore it is an effective mitigation against attacks in the real world.

Chapter 6

Future Work

This chapter examines directions for future work that we believe to be realistic. First, we discuss two main directions that we see for the photonic side channel. Then, we sketch ideas for future work regarding higher-order fault attacks on pairing computations.

6.1 The Photonic Side Channel

We see two main directions for future work: first, the photonic side channel has to be explored in more detail. To uncover its full potential, its advantages and the requirements for a successful attack have to be fully understood. Second, we have to restrict this potential by developing effective and efficient countermeasures. We have to carefully examine all existing countermeasures against other side channels and test whether or not they are also effective against photonic side channel attacks. Then, countermeasures tailored to specifically mitigate photonic emission analysis have to be developed.

Exploring the Full Attack Potential

We attacked AES-128 encryption in this work. All three presented distinguishers for DPEA need considerably less than 256 input data to reveal the entire 128-bit key if the captured traces carry a good signal. Hence, a DPEA with the 256 chosen plaintexts in the manner described in Chapter 4 can be directly transferred to AES-192 and AES-256 and longer keys, respectively, and will also recover the entire secret key. Thus, the attack does not have to be adapted for longer keys. However, as it is the case for other side channels, we can also attack the decryption process with photonic emission analysis. The first three operations of AES decryption are the inverses of `AddRoundKey`, `ShiftRows`, and `SubBytes`. Thus, both SPEA and DPEA as explained in this work can be directly applied and the application of PEA to the decryption process is straight forward. Nevertheless, this should be verified in practice.

It is more important, however, to target real-world implementations, both in software and in hardware. The ATXMega128A1 integrates both AES and DES crypto modules, which are relevant targets. The current DPA contest, DPA contest v4, is also interesting [173]. It includes two masked implementations of AES: a

masked implementation of AES-256 and an improved masked implementation of AES-128, both on an Atmel ATMega-163 smartcard.

Higher-order attacks are also a promising research direction. These can not only defeat countermeasures like randomization and masking, but exploit several sources of leakage simultaneously and in conjunction will yield more efficient attacks. In the presented SPEA, additional measurements of other points of leakage make it feasible to further reduce the set of potential key candidates also for an aligned S-Box. It is feasible, for example, to exploit the leakage of the column decode logic to reveal more information about the address of the accessed memory content. Moreover, measuring emissions of SubBytes operations in further rounds will also help to decrease the number of remaining key candidates. If additional measurements can be captured in a reasonable amount of time, the attack could even be implemented as an unknown-plaintext attack, as demonstrated for a cache timing attack [83].

In Section 4.1.2, we already sketched how an SPEA can be conducted against other algorithms such as LED and PRESENT as well as algorithms for binary exponentiation and multiplication. Currently, we are also exploring the potential for attacking COMP128 (v2/ v3), an algorithm used in the mobile phone standard Global System for Mobile Communications (GSM). We assume that most algorithms that use an S-Box are particularly susceptible to photonic side channel attacks. Therefore, they all have to be analyzed. In this context, we also suggest to develop an SPEA against the AES software implementation that uses precomputed tables, as it is used in OpenSSL. This implementation was already the target of several side channel attacks, e.g., by Bernstein [22]. Moreover, all relevant encryption algorithms have to be analyzed with respect to their risk of being compromised by a PEA, both symmetric and asymmetric, and also other cryptographic algorithms such as those for signatures.

As it was pointed out in Section 4.2.2, the role of the distinguisher seems to be not too important concerning the generic distinguishers that we used. However, the noise distribution is an important factor [91]. We assume that the leakage distribution may also be a relevant factor. As it was pointed out by W. Schindler during a conversation, considering the Poisson distribution of the photonic leakage might increase the efficiency of differential analyses, especially using the stochastic approach. Consequently, it should be examined if there are other distinguishers especially well suited for the analysis of photonic emission, or if it is possible to adapt distinguishers to the specifics of this side channel. Therefore, optimal distinguishers for different scenarios should be tested [91]. Analyses based on the linear regression promise to be both efficient and flexible and should therefore be examined in more detail [181]. Moreover, not only generic, but also profiled attacks, which exploit information about the leakage distribution, should be conducted.

Independent of the chosen distinguisher, more analyses of the efficiency of this side channel have to be conducted. What is the minimum number of traces for a successful SPEA or DPEA when the attacker can use chosen input data? How does this number increase when we do not use chosen messages, but random known

plaintexts? How efficient can an analysis be in the case that the attacker does not know the input data? Different attack scenarios and leakage models have to be evaluated to gain more insights into the limitations of the photonic side channel.

For a side channel attack which aims at further improving the design and at implementing architecture-specific countermeasures, the characterization of the leakage is interesting as well. Regarding the protection of cryptographic implementations and to guarantee a desired level of side channel resistance, leakage characterization is of the upmost importance. While our work on a comprehensive constructiveness analysis is still in process, we invite the research community to investigate how photonic emission analysis can help to gain knowledge about the attacked device and its implementation. One part of the leakage characterization is to examine leakage differences between 0-1 and 1-0 transitions, one of which has stronger emissions than the other one if n-type and p-type transistors have the same size. If the attacker knows the circuit that she attacks in detail, she also knows which transition should be stronger. This knowledge might help her to improve the analysis and to gain insights about the construction of the device.

Apart from these ideas concerning the cryptographic side channel, the research community has to consider further attack scenarios which arise from the detailed spatial orientation that photonic emission analysis provides and the high spatial resolution that it offers. This basic methodology can be applied to many different attack vectors. The first result in this direction, a physical characterization of arbiter Physically Unclonable Functions (PUFs), has already been published [166].

Developing Countermeasures

Regarding the countermeasures, we first suggest to test all countermeasures that exist for other side channels to find out which of these are effective and which can easily be circumvented with a photonic side channel attack. Thus, the specific characteristics of PEA have to be considered.

Most countermeasures presented in Sections 4.1 and 4.2 have been sketched, but have not been described in detail. The research community, in academia as well as in industry, needs to investigate these ideas and to do further research in this direction. Useful countermeasures against SPEA and DPEA can affect the hardware as well as the implementation of a cipher. If they are too expensive to be applied to all rounds of a block cipher, especially the first and the last rounds have to be protected.

From the view of an attacker, the need for averaging thousands of measurements to yield a sufficient signal is an obvious disadvantage. However, this helps chip designers and software engineers to mitigate such an analysis by properly implementing randomized countermeasures.

The photonic side channel offers the huge advantage that leakage can be captured with a high spatial resolution. Effectively, every transistor exhibiting data-dependent behavior becomes a potential target. Therefore, layout design rules must

address this threat. One of the main targets of the development of countermeasures should be to prevent the processing of key-dependent data at a single spot such as the driving inverters that we attacked with the presented DPEA. Under the assumption that the attacker uses only a single APD detector, the parallelization of sensitive operations will considerably increase the workload of an attack.

Many standard industry countermeasures do not prevent photonic emission attacks. Metal layers, shields and meshes generally only protect against attacks from the frontside, and thus do not prevent the photonic emission attacks presented in this work. Hence, to prevent photonic emission analysis altogether, countermeasures must be developed to shield photonic emissions from reaching the observer from the backside, although such countermeasures would make the ICs more expensive to produce. An active shield or mesh on the backside of an IC will prohibit any optical emission attack. A single metal layer on the backside can trap all photons generated within the chip. As soon as this is incorporated, it has to be protected against being removed. This can be ensured by active integrity checks. Backside analysis can also be impeded by doping the silicon layer. Doping the silicon layer implies that it is no longer transparent in the spectral range above 1 µm. Thus, the silicon has to be polished also for the emission measurements with the APD, the detector which is necessary for temporal analysis. Furthermore, sensitive data should only be processed very deep inside the device to promote the goal of preventing relevant photonic emissions from reaching the observer.

The assumptions which underlie these proposed countermeasures will not be valid forever. Therefore, the lifecycle of an IC generation has always to be considered when countermeasures are developed and integrated. When designing implementations with such countermeasures, the continuous development in optical technologies also has to be considered. Advances in optical technologies will lead to better photon detection mechanisms, which will help attackers to circumvent some countermeasures.

6.2 Fault Attacks against Pairing-Based Cryptography

The higher-order fault attack presented in this work is the first published practical fault attack against cryptographic pairings. Hence, the full attack potential of fault attacks against pairing computations has not been fully explored yet. Furthermore, all theoretical fault attacks have not considered cryptographic protocols, but only the computation of a single pairing. Therefore it has to be examined how these attacks can be applied to concrete applications of pairings. Countermeasures to prevent such attacks are discussed in Chapter 5.

Exploring the Full Attack Potential

The first fault attack on elliptic curve cryptosystems consisted in modifying the coordinates of a point so that the new point would not be on the original curve, but on another one [24]. To the best of our knowledge, this concept has not been

transferred to PBC yet, but might be an interesting approach. This attack might even be possible without any fault induction, as the authors of [24] already stated: if the DUA does not explicitly check whether or not the input points are on the specified curve, a malicious user can just input a point with the desired properties. Thus, it should be investigated if an attacker can benefit from a public input point which lies on another curve.

In the presented attack, the final exponentiation was completely removed by an instruction skip. This was possible since the RELIC code comprises a function call to the final exponentiation in our setup. However, as explained in Section 5.4, in other realistic implementations this might not be the case due to code optimizations. This attack vector does neither exist in our implementation if one compiles the RELIC library with optimization level `-O2` instead of `-O1`. Then, the compiler replaces the function call to the final exponentiation with inline code, i.e., the call is replaced by the code in the body of the program. Thus, our approach of completely skipping the final exponentiation with a single instruction skip is blocked. Therefore, it is necessary to figure out how the final exponentiation can be attacked in the case that there is no dedicated function call to this operation. Then, the final exponentiation most probably cannot completely be removed, but only manipulated. Therefore, an improved mathematical analysis is also necessary for the exponentiation inversion in future attacks on more realistic scenarios. Such an improved algebraic analysis can build upon [175].

We generated the necessary instruction skips by means of clock glitching. Other techniques such as laser fault attacks can lead to the same result. A double-fault laser attack could already successfully be launched against a protected RSA-CRT implementation [170]. Hence, it has to be investigated which kinds of physical attacks might lead to the same effects on a cryptographic algorithm.

Targeting Cryptographic Protocols

Our attack does not target a cryptographic protocol that is based on pairings, but only the computation of a single pairing operation. This allows to control the input points and to access the output of this operation directly. In realistic applications of pairings, however, pairings are used in cryptographic protocols such as IBE, Attribute-Based Encryption (ABE) [144], and oblivious transfer protocols [43].

All fault attacks against pairings assume that the pairing has two input points, one of which is public and one of which is secret key material. It is assumed that the computation can be repeated several times with the same parameters. This is what we assume in our attack described in Section 5.2 and what others assume in their work, e.g., [63, 133]. During a pairing-based cryptographic protocol, however, often the inputs to the pairing are not constant if, for example, the same plaintext is encrypted several times for the same receiver. During the decryption in the FullIdent scheme, the input point of the pairing which is assumed to be public is randomly generated during encryption [38]. Hence, the assumption that the attacker can access several repetitions of exactly the same decryption operation does not hold.

Moreover, for many pairing-based protocols, the output of the pairing cannot be directly accessed by an attacker. In the FullIdent scheme, a bilinear pairing is used during encryption and decryption [38]. From an attacker's perspective, the decryption is a more interesting target, since the encryption process does not use secret key material. During the decryption process, the pairing uses secret key material. The result of this pairing computation, however, is not the output of the decryption, but only the input for a cryptographic hash function, the result of which is also further processed. Thus, it should also be investigated how an experienced attacker can obtain the result of the pairing in a real attack despite subsequent further operations.

Very recently, it was shown that only a few full cryptographic protocols based on PBC actually succumb the fault attacks described in [133] and [180], see [46]. The effectiveness of these attacks was presented against three protocols [40, 43, 78]. It was shown that most other protocols with one secret input point and one public input point do not release the result of a pairing. It was further shown that the attacks can only be applied to these vulnerable protocols when they are implemented with symmetric pairings, i.e., Type 1 pairings. Otherwise, a necessary embedding from the message space to the image set of the pairing does not exist.

Moreover, in most fault attacks it is assumed that a single pairing computation has to be attacked. In order to ensure leakage resilience, however, protocols with shared keys and accordingly multiple pairing computations are developed [74]. At first appearance and under the assumption that an attacker can only launch single-fault attacks, these are also more secure against active physical attacks. With our setup, however, multiple faults can easily be generated and hence, also multiple pairings in a single decryption operation can be attacked.

Hence, the most interesting direction for further research is to develop theoretical and practical fault attacks against cryptographic protocols which are used in real applications, and to develop and implement countermeasures against these kinds of attacks.

Chapter 7

Conclusion

This thesis demonstrates on the basis of two physical attacks against cryptographic systems that the complexity of physical attacks should not be overestimated. Overestimating the physical attack complexity leads to unprotected devices once the estimation of the complexity proves wrong and the attack can be realized.

First, we presented photonic emission as cryptographic side channel. This side channel was not perceived as a serious threat when it was first published due to its high cost. We explained the theory of Simple Photonic Emission Analysis and Differential Photonic Emission Analysis and showed successful photonic side channel attacks against AES based on a low-cost measurement setup.

Second, we presented a practical second-order fault attack on the eta pairing η_n. Fault attacks on pairing computations were assumed to be unrealistic due to the need for several precise fault insertions during a single computation. We categorized the effects of fault attacks against the Miller Algorithm and described the cryptanalysis of three examples for second-order fault attacks against different pairings, which can all be conducted with a glitching platform that we developed.

7.1 The Photonic Side Channel

Once the low-cost setup was developed, both SPEA and DPEA proved to be a powerful tool and thus showed that photonic side channel attacks pose a serious risk to modern security ICs. We developed the theory of Simple Photonic Emission Analysis and Differential Photonic Emission Analysis, analogous to SPA, DPA, EMA, and DEMA. We presented a practical SPEA and a practical DPEA targeting the first SubBytes operation of AES on two different microcontrollers. We compared several distinguishers for the differential analyses and confirmed that they are similar in terms of efficiency, as it was shown for other side channels as well. We explained that photonic side channel attacks are not AES-specific but can also be applied to other cryptographic algorithms. Given the low-cost setup and the methodology of SPEA and DPEA, the photonic side channel complements the cryptanalytic tools for attacking cryptography.

Since photonic side channel attacks can be used to exploit photonic emissions even of selected components of the attacked device, they have to be addressed by chip designers, chip manufacturers, and software engineers. All involved parties do not only have to consider global side channels such as power analysis but also all local attack vectors.

Since the photonic side channel poses a threat to unprotected implementations, powerful hardware and software countermeasures that directly target the leakage from photonic emissions have to be developed. This work presents several solutions. Countermeasures developed to mitigate power analysis can also hinder photonic emission analysis. However, the extraordinary spatial resolution of photonic emission and the resulting large number of potentially leaking targets offer attack vectors that have not been considered in the development of countermeasures against power analysis attacks so far. Moreover, since emission images allow for a functional understanding of the DUA, some countermeasures can be easily circumvented by selecting a different area on the chip. Thus, PEA-specific solutions have to be developed. These can be measures on the technology level, such as absorbing dotant profiles or substrate treatment to hinder photonic emissions from reaching the observer altogether. Novel standard cell layouts that reduce data-dependent emission also make the physical attack more complex. In addition to technical solutions, algorithmic modifications such as PEA-specific masking have to be developed and implemented.

When designing implementations with such countermeasures, the continuous development in optical technologies has to be considered: for the first results on the photonic side channel, millions of measurements and many hours of signal acquisition were needed for a successful attack. Latest research, however, shows that several thousand measurements, taken in a few minutes, are sufficient.

7.2 Fault Attacks against Pairing-Based Cryptography

Although fault attacks against PBC have been described for more than ten years, no practical experiments have been reported before we published our results. These results prove that practical attacks pose a serious threat to pairings. Our attack did not only include a single fault but even two faults within a single pairing computation. Two faults are necessary in most scenarios but have been regarded as unrealistic due to the complexity of inserting several precise faults during a single computation. We demonstrated a practical double-fault attack on the eta pairing η_n. We used the first fault to modify the computation of the Miller function and completely skipped the final exponentiation with the second fault. We categorized the effects of fault attacks against the Miller Algorithm and described two more examples of second-order fault attacks against different pairings.

Our setup allows an attacker to induce clock glitches which provoke the skipping of chosen instructions. With this system, all theoretical fault attacks that have been proposed against PBC so far can be successfully carried out in practice.

Since we demonstrated that fault attacks against PBC pose a real threat against cryptographic devices, countermeasures have to be developed to secure these devices. We can skip any instruction with our setup. Therefore, all pairing algorithms have to be investigated with regard to their susceptibility towards instruction skip attacks. Each attack vector has to be prevented. When developing countermeasures, not only clock glitches and instruction skips have to be considered. Higher-order fault

attacks have also been conducted with other techniques such as lasers [170]. Laser attacks allow for instruction skips as well but are also very accurate in targeting a particular variable. Hence, they can also be used to carry out attacks on pairing computations by, e.g., modifying the Miller bound.

Our setup allows for much more than just double-fault attacks. Thus, we believe that future developments and improvements of practical attacks should be considered when implementing countermeasures.

7.3 Advice for Cryptographers

This work shows that cryptography should not rely on physical attack complexity. We presented two practical attacks - a side channel attack and a fault attack - both of which have been considered unrealistic due to the complexity of their realization. Presenting these successful attacks reveals that attacks that seem infeasible today can soon become reality.

The attacks presented in this work are not the only examples for threats that were initially underestimated. Only recently two more assessments of physical complexity proved to be wrong: the acoustic side channel and cloning of PUFs. The acoustic side channel, which exploits variations in the sound generated by a computer, was known for ten years, but it has not been taken seriously until practical results were presented in 2013 [77]. Measuring acoustic emanations with a sufficient signal strength was considered impossible with a good quality due to the low bandwidth and the low emission strength of this side channel. When the acoustic side channel was experimentally demonstrated, however, only common software and hardware were used. PUFs, on the other hand, rely per definition on their physical unclonability. Unpredictability and unclonability are the main requirements of PUFs [76]. They became a vibrant field of research during the last years and promised to be "a means of building secure smartcards" due to their assumed resistance to physical attacks [76]. Very recently, however, it was shown that SRAM PUFs are not unclonable [88] and the same setup that we used for the photonic side channel attacks was used to prove that all arbiter PUFs can be completely and linearly characterized by means of PEA [166].

Hence, cryptographic devices and their implementations should not only be secured against contemporary attacks but also against those which seem infeasible today. Cryptographers and engineers should not only react to novel attacks but anticipate these threats and implement countermeasures as soon as a novel attack has been described theoretically. It was conjectured that "countermeasures can neither be formalized nor tested without a sound understanding of attacks" [113]. This does not imply, however, that the attack has to be practically demonstrated before mitigations can be developed. As soon as it is known how an attack works in theory, countermeasures can be developed. It was known for some years that photonic emissions can provide a highly spatially resolved side channel, and most of the countermeasures that we suggested for the photonic side channel could have

been developed already in light of this threat. It was also known that the final exponentiation protects cryptographic pairings against simple fault attacks, but no serious effort was put into the protection of pairings in the case that an attacker circumvents the final exponentiation.

Cryptographic devices are not protected against attacks which are known in theory but have not been conducted yet. Cryptographers should outclass attackers by protecting their devices and data as soon as a threat is theoretically known. When an attack is known theoretically, yet perceived as irrelevant due to the attack complexity, countermeasures should still be developed to thwart the attack. Cryptographers should not rely on physical attack complexity and wait until a practical attack convinces them of the threat. Instead they should anticipate the threat and develop mitigation techniques in a proactive manner.

Acronyms

ABE Attribute-Based Encryption
AES Advanced Encryption Standard
APD Avalanche Photo Diode

CCD Charge-Coupled Device
CMOS Complementary Metal-Oxide-Semiconductor
CPU Central Processing Unit

DDK Die Datenkrake
DEMA Differential Electromagnetic Analysis
DES Data Encryption Standard
DFA Differential Fault Analysis
DLP Discrete Logarithm Problem
DoM Difference of Means
DPA Differential Power Analysis
DPEA Differential Photonic Emission Analysis
DRAM Dynamic Random-Access Memory
DSA Digital Signature Algorithm
DUA Device Under Attack

ECC Elliptic Curve Cryptography
ECDLP Elliptic Curve Discrete Logarithm Problem
ECDSA Elliptic Curve Digital Signature Algorithm
EM electromagnetic
EMA Electromagnetic Analysis

FIFO First In - First Out
FPGA Field Programmable Gate Array

GSM Global System for Mobile Communications
GSR Global Success Rate

HD Hamming Distance
HRP Hidden Root Problem
HW Hamming Weight

IBC Identity-Based Cryptography
IBE Identity-Based Encryption
IC Integrated Circuit
IR infrared

LED Light Encryption Device
LSB Least Significant Byte
lsb Least Significant Bit

MOSFET Metal-Oxide-Semiconductor Field-Effect Transistor
msb Most Significant Bit

NIR near-infrared

PBC Pairing-Based Cryptography
PCB Printed Circuit Board
PEA Photonic Emission Analysis
PICA Picosecond Imaging Circuit Analysis
PKG Private Key Generator
PMT Photo Multiplier Tube
PoC Proof of Concept
PSR Partial Success Rate
PUF Physically Unclonable Function

RFID Radio-Frequency Identification
RISC Reduced Instruction Set Computer
RPI Random Process Interrupt

SEMA Simple Electromagnetic Analysis
SNR Signal-to-Noise Ratio
SPA Simple Power Analysis
SPEA Simple Photonic Emission Analysis
SRAM Static Random-Access Memory
SSPD Superconducting Single Photon Detector

TDC Time-to-Digital Converter

VIS visible spectrum

WSN Wireless Sensor Network

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] ATmega48A/PA/88A/PA/168A/PA/328/P Complete datasheet. http://www.atmel.com/PRODUCTS/MICROCONTROLLERS/AVR/?tab=documents. [accessed 2014/11/10].

[2] ATxmega64A1/128A1 Preliminary datasheet. http://www.atmel.com/devices/atxmega128a1.aspx. [accessed 2014/04/09].

[3] Cortex-M3 Processor. www.arm.com/products/processors/cortex-m/cortex-m3.php. [accessed 2014/04/09].

[4] Die Datenkrake. datenkrake.org. [accessed 2014/04/09].

[5] ODROID-U2 Product Page. http://hardkernel.com/main/products/prdt_info.php?g_code=G135341370451. [accessed 2014/05/29].

[6] Python Programming Language Homepage. http://www.python.org. [accessed 2014/05/15].

[7] G. Adj, A. Menezes, T. Oliveira, and F. Rodríguez-Henríquez. Computing Discrete Logarithms in $F_{3^{6*137}}$ and $F_{3^{6*163}}$ Using Magma. *IACR Cryptology ePrint Archive, Report 2014/057*, 2014.

[8] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM Side-Channel(s). In B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer, 2002.

[9] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM Side-Channel(s): Attacks and Assessment Methodologies. http://web.cs.jhu.edu/~astubble/600.412/s-c-papers/em.pdf, 2002. [accessed 2014/10/14].

[10] S. Ali and D. Mukhopadhyay. Protecting Last Four Rounds of CLEFIA is Not Enough against Differential Fault Analysis. *IACR Cryptology ePrint Archive, Report 2012/286*, 2012.

[11] D. F. Aranha, P. S. L. M. Barreto, P. Longa, and J. E. Ricardini. The Realm of the Pairings. In *Selected Areas in Cryptography*, volume 8282 of *Lecture Notes in Computer Science*, pages 3–25. Springer Berlin Heidelberg, 2013.

[12] D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. http://code.google.com/p/relic-toolkit/. [accessed 2014/04/09].

[13] K. Bae, S. Moon, and J. Ha. Instruction Fault Attack on the Miller Algorithm in a Pairing-Based Cryptosystem. In L. Barolli, I. You, F. Xhafa, F. Leu, and H. Chen, editors, *Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2013*, pages 167–174. IEEE, 2013.

[14] J. Balasch, S. Faust, B. Gierlichs, and I. Verbauwhede. Theory and Practice of a Leakage Resilient Masking Scheme. In X. Wang and K. Sako, editors, *Advances in Cryptology - ASIACRYPT 2012, 18th International Conference on the Theory and Application of Cryptology and Information Security*, volume 7658 of *Lecture Notes in Computer Science*, pages 758–775. Springer Berlin Heidelberg, 2012.

[15] J. Balasch, B. Gierlichs, and I. Verbauwhede. An In-depth and Black-Box Characterization of the Effects of Clock Glitches on 8-bit MCUs. In L. Breveglieri, S. Guilley, I. Koren, D. Naccache, and J. Takahashi, editors, *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 105–114. IEEE, 2011.

[16] S. Banik and S. Maitra. A Differential Fault Attack on MICKEY 2.0. In G. Bertoni and J.-S. Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013*, pages 215–232. Springer Berlin Heidelberg, 2013.

[17] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The Sorcerer's Apprentice Guide to Fault Attacks. *IACR Cryptology ePrint Archive, Report 2004/100*, 2004.

[18] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic. In P. Q. Nguyen and E. Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2014.

[19] P. S. L. M. Barreto, S. D. Galbraith, C. O'Eigeartaigh, and M. Scott. Efficient Pairing Computation on Supersingular Abelian Varieties. *Des. Codes Cryptography*, 42(3):239–271, 2007.

[20] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In M. Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer, 2002.

[21] G. Bascoul, P. Perdu, A. Benigni, S. Dudit, G. Celi, and D. Lewis. Time Resolved Imaging: From Logical States to Events, a New and Efficient Pattern Matching Method for VLSI Analysis. *Microelectronics Reliability*, 51(9-11):1640–1645, 2011.

[22] D. J. Bernstein. Cache-Timing Attacks on AES. http://cr.yp.to/papers.html#cachetiming, 2004. [accessed 2014/12/15].

[23] D. J. Bernstein, J. Buchmann, and E. Dahmen. *Post Quantum Cryptography*. Springer Publishing Company, Incorporated, first edition, 2008.

[24] I. Biehl, B. Meyer, and V. Müller. Differential Fault Attacks on Elliptic Curve Cryptosystems. In M. Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference*, volume 1880 of *Lecture Notes in Computer Science*, pages 131–146. Springer, 2000.

[25] E. Biham. New Types of Cryptanalytic Attacks Using Related Keys. *J. Cryptology*, 7:229–246, 1994.

[26] E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In A. Menezes and S. A. Vanstone, editors, *Advances in Cryptology - CRYPTO 1990, 10th Annual International Cryptology Conference*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer-Verlag, 1991.

[27] E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In B. S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO 1997, 17th Annual International Cryptology Conference*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer Berlin Heidelberg, 1997.

[28] A. Biryukov and D. Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In M. Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.

[29] I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic Curves in Cryptography*, volume 265. Cambridge University Press, 1999.

[30] I. F. Blake, G. Seroussi, and N. P. Smart, editors. *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 2005.

[31] J. Blömer, R. G. da Silva, P. Günther, J. Krämer, and J.-P. Seifert. A Practical Second-Order Fault Attack against a Real-World Pairing Implementation. In A. Tria and D. Choi, editors, *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2014, Busan, South Korea*, pages 123–136. IEEE Computer Society, 2014.

[32] J. Blömer, J. Guajardo, and V. Krummel. Provably Secure Masking of AES. In H. Handschuh and M. A. Hasan, editors, *Selected Areas in Cryptography, 11th International Workshop, SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2004.

[33] J. Blömer, P. Günther, and G. Liske. Improved Side Channel Attacks on Pairing Based Cryptography. In E. Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - 4th International Workshop, COSADE 2013*, volume 7864 of *Lecture Notes in Computer Science*, pages 154–168. Springer Berlin Heidelberg, 2013.

[34] J. Blömer, P. Günther, and G. Liske. Tampering Attacks in Pairing-Based Cryptography. In A. Tria and D. Choi, editors, *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2014, Busan, South Korea*, pages 1–7. IEEE Computer Society, 2014.

[35] J. Blömer, M. Otto, and J.-P. Seifert. Sign Change Fault Attacks on Elliptic Curve Cryptosystems. In L. Breveglieri, I. Koren, D. Naccache, and J.-P. Seifert, editors, *FDTC*, volume 4236 of *Lecture Notes in Computer Science*, pages 36–52. Springer, 2006.

[36] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[37] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer Berlin Heidelberg, 1997.

[38] D. Boneh and M. K. Franklin. Identity-Based Encryption from the Weil Pairing. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.

[39] W. Bosma, J. Cannon, and C. Playoust. The Magma Algebra System. I. The User Language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997.

[40] X. Boyen, Q. Mei, and B. Waters. Direct Chosen Ciphertext Security from Identity-Based Techniques. *IACR Cryptology ePrint Archive, Report 2005/288*, 2005.

[41] E. Brier, B. Chevallier-Mames, M. Ciet, and C. Clavier. Why One Should Also Secure RSA Public Key Elements. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 324–338. Springer Berlin Heidelberg, 2006.

[42] D. Brumley and D. Boneh. Remote Timing Attacks Are Practical. In *Proceedings of the 12th USENIX Security Symposium, Washington, D.C., USA, 2003*. USENIX Association, 2003.

[43] J. Camenisch, G. Neven, and A. Shelat. Simulatable Adaptive Oblivious Transfer. In M. Naor, editor, *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 573–590. Springer Berlin Heidelberg, 2007.

[44] S. Chang, H. Hong, E. Lee, and H.-S. Lee. Reducing Pairing Inversion to Exponentiation Inversion Using Non-Degenerate Auxiliary Pairing. *IACR Cryptology ePrint Archive, Report 2013/313*, 2013.

[45] S. Chari, J. R. Rao, and P. Rohatgi. Template Attacks. In B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[46] S. Chatterjee, K. Karabina, and A. Menezes. Fault Attacks on Pairing-Based Protocols Revisited. *IEEE Trans. Computers*, 64(6):1707–1714, 2015.

[47] H. Choukri and M. Tunstall. Round Reduction Using Faults. In L. Breveglieri and I. Koren, editors, *2005 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 13 – 24, 2005.

[48] C.-K. Chu, J. K. Liu, J. Zhou, F. Bao, and R. H. Deng. Practical ID-Based Encryption for Wireless Sensor Network. In D. Feng, D. A. Basin, and P. Liu, editors, *ACM Symposium on Information, Computer and Communications Security, ASIA CCS*, pages 337–340. ACM, 2010.

[49] A. G. Chynoweth and K. G. McKay. Photon Emission from Avalanche Breakdown in Silicon. *Phys. Rev.*, 102:369–376, 1956.

[50] M. Ciet and M. Joye. Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults. *Des. Codes Cryptography*, 36(1):33–43, 2005.

[51] C. Clavier, J.-S. Coron, and N. Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 13–48. Springer Berlin / Heidelberg, 2000.

[52] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman & Hall/CRC, second edition, 2012.

[53] J.-S. Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer Berlin Heidelberg, 1999.

[54] J.-S. Coron and I. Kizhvatov. An Efficient Method for Random Delay Generation in Embedded Software. In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 2009.

[55] J.-C. Courrège, B. Feix, and M. Roussellet. Simple Power Analysis on Exponentiation Revisited. In D. Gollmann, J.-L. Lanet, and J. Iguchi-Cartigny, editors, *Smart Card Research and Advanced Application - CARDIS 2010*, volume 6035 of *Lecture Notes in Computer Science*, pages 65–79. Springer Berlin Heidelberg, 2010.

[56] N. Courtois and L. Goubin. An Algebraic Masking Method to Protect AES against Power Attacks. In D. Won and S. Kim, editors, *International Conference on Information Security and Cryptology - ICISC 2005*, volume 3935 of *Lecture Notes in Computer Science*, pages 199–209. Springer, 2005.

[57] D. A. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, (Undergraduate Texts in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, third edition, 2007.

[58] J. Daemen and V. Rijmen. *The Design of Rijndael: AES – the Advanced Encryption Standard*. Springer Berlin / Heidelberg, 2002.

[59] J. Di-Battista, J.-C. Courrege, B. Rouzeyre, L. Torres, and P. Perdu. When Failure Analysis Meets Side-Channel Attacks. In S. Mangard and F.-X. Standaert, editors, *Cryptographic Hardware and Embedded Systems - CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 188–202. Springer Berlin/Heidelberg, 2011.

[60] J. Doget, E. Prouff, M. Rivain, and F.-X. Standaert. Univariate Side Channel Attacks and Leakage Modeling. *J. Cryptographic Engineering*, 1(2):123–144, 2011.

[61] E. Dottax, C. Giraud, M. Rivain, and Y. Sierra. On Second-Order Fault Analysis Resistance for CRT-RSA Implementations. In O. Markowitch, A. Bilas, J. Hoepman, C. J. Mitchell, and J. Quisquater, editors, *Workshop in Information Security Theory and Practice, WISTP 2009*, volume 5746 of *Lecture Notes in Computer Science*, pages 68–83. Springer, 2009.

[62] P. Egger, M. Grutzner, C. Burmer, and F. Dudkiewicz. Application of Time Resolved Emission Techniques within the Failure Analysis Flow. *Microelectronics Reliability*, 47(9-11):1545–1549, 2007.

[63] N. El Mrabet. What about Vulnerability to a Fault Attack of the Miller's Algorithm During an Identity Based Protocol? In *Proceedings of the 3rd International Conference and Workshops on Advances in Information Security and Assurance*, ISA '09, pages 122–134. Springer Berlin Heidelberg, 2009.

[64] N. El Mrabet, G. Di Natale, and M. Flottes. A Practical Differential Power Analysis Attack against the Miller Algorithm. In *PRIME 2009 - 5th Conference on Ph.D. Research in Microelectronics and Electronics, Circuits and Systems Magazine*, 2009.

[65] N. El Mrabet, D. Page, and F. Vercauteren. Fault Attacks on Pairing-Based Cryptography. In M. Joye and M. Tunstall, editors, *Fault Analysis in Cryptography*. Springer Berlin Heidelberg, 2012.

[66] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 1968.

[67] J. Ferrigno and M. Hlaváč. When AES Blinks: Introducing Optical Side Channel. *Information Security, IET*, 2(3):94 –98, 2008.

[68] A. M. Fiskiran and R. B. Lee. Fast Parallel Table Lookups to Accelerate Symmetric-Key Cryptography. In *International Symposium on Information Technology: Coding and Computing (ITCC 2005)*, pages 526–531. IEEE Computer Society, 2005.

[69] P. Fouque, R. Lercier, D. Réal, and F. Valette. Fault Attack on Elliptic Curve Montgomery Ladder Implementation. In L. Breveglieri, S. Gueron, I. Koren, D. Naccache, and J.-P. Seifert, editors, *Fifth International Workshop on Fault Diagnosis and Tolerance in Cryptography, 2008, FDTC 2008, Washington, DC, USA*, pages 92–98. IEEE Computer Society, 2008.

[70] D. Freeman, M. Scott, and E. Teske. A Taxonomy of Pairing-Friendly Elliptic Curves. *J. Cryptology*, 23(2):224–280, 2010.

[71] S. D. Galbraith. *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012.

[72] S. D. Galbraith, F. Hess, and F. Vercauteren. Aspects of Pairing Inversion. *IEEE Transactions on Information Theory*, 54(12):5719–5728, 2008.

[73] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for Cryptographers. *Discrete Appl. Math.*, 156(16):3113–3121, 2008.

[74] D. Galindo, J. Großschädl, Z. Liu, P. K. Vadnala, and S. Vivek. Implementation and Evaluation of a Leakage-Resilient ElGamal Key Encapsulation Mechanism. *IACR Cryptology ePrint Archive, Report 2014/835*, 2014.

[75] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[76] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon Physical Random Functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 148–160. ACM, 2002.

[77] D. Genkin, A. Shamir, and E. Tromer. RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. In J. A. Garay and R. Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 444–461. Springer, 2014.

[78] C. Gentry. Practical Identity-Based Encryption Without Random Oracles. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer, 2006.

[79] S. Ghosh, D. Mukhopadhyay, and D. R. Chowdhury. Fault Attack and Countermeasures on Pairing Based Cryptography. *International Journal of Network Security*, 12(1):21–28, 2011.

[80] R. Gomes da Silva. Practical Analysis of Embedded Microcontrollers against Clock Glitching Attacks. Bachelor's thesis, Technische Universität Berlin, 2014.

[81] R. Granger, T. Kleinjung, and J. Zumbrägel. Breaking '128-bit Secure' Supersingular Binary Curves (Or How to Solve Discrete Logarithms in $\mathbb{F}_{2^{4\cdot 1223}}$ and $\mathbb{F}_{2^{12\cdot 367}}$). In J. A. Garay and R. Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA 2014, Proceedings, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 126–145. Springer, 2014.

[82] S. Gueron and J.-P. Seifert. Is It Wise to Publish Your Public RSA Keys? In L. Breveglieri, I. Koren, D. Naccache, and J.-P. Seifert, editors, *2006 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, volume 4236 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[83] D. Gullasch, E. Bangerter, and S. Krenn. Cache Games – Bringing Access-Based Cache Attacks on AES to Practice. In *Security and Privacy, 2011 IEEE Symposium on*, pages 490 –505, 2011.

[84] J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw. The LED Block Cipher. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011.

[85] S. Hajra and D. Mukhopadhyay. SNR to Success Rate: Reaching the Limit of Non-Profiling DPA. *IACR Cryptology ePrint Archive, Report 2013/865*, 2013.

[86] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest We Remember: Cold-Boot Attacks on Encryption Keys. *Commun. ACM*, 52(5):91–98, 2009.

[87] D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., 2003.

[88] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert. Cloning Physically Unclonable Functions. In *Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on*, pages 1–6, 2013.

[89] F. Hess. Efficient Identity Based Signature Schemes Based on Pairings. In K. Nyberg and H. M. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer, 2002.

[90] A. Heuser, M. Kasper, W. Schindler, and M. Stöttinger. A New Difference Method for Side-Channel Analysis with High-Dimensional Leakage Models. In O. Dunkelman, editor, *Topics in Cryptology - CT-RSA 2012*, volume 7178 of *Lecture Notes in Computer Science*, pages 365–382. Springer, 2012.

[91] A. Heuser, O. Rioul, and S. Guilley. Good Is Not Good Enough - Deriving Optimal Distinguishers from Communication Theory. In L. Batina and M. Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 55–74. Springer, 2014.

[92] J. Heyszl. *Impact of Localized Electromagnetic Field Measurements on Implementations of Asymmetric Cryptography*. Dissertation, Technische Universität München, 2013.

[93] J. Heyszl, S. Mangard, B. Heinz, F. Stumpf, and G. Sigl. Localized Electromagnetic Analysis of Cryptographic Implementations. In O. Dunkelman, editor, *Topics in Cryptology — CT-RSA 2012*, volume 7178 of *Lecture Notes in Computer Science*, pages 231–244. Springer Berlin / Heidelberg, 2012.

[94] A. Joux. A One Round Protocol for Tripartite Diffie-Hellman. In W. Bosma, editor, *Proceedings of the 4th International Symposium on Algorithmic Number Theory*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer Berlin Heidelberg, 2000.

[95] M. Joye, P. Paillier, and B. Schoenmakers. On Second-Order Differential Power Analysis. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005.

[96] L. Judge, M. Cantrell, C. Kendir, and P. Schaumont. A Modular Testing Environment for Implementation Attacks. *ASE/IEEE International Conference on BioMedical Computing (BioMedCom)*, pages 86–95, 2012.

[97] N. Kanayama and E. Okamoto. Approach to Pairing Inversions Without Solving Miller Inversion. *IEEE Transactions on Information Theory*, 58(2):1248–1253, 2012.

[98] J. Kash and J. Tsang. Dynamic Internal Testing of CMOS Circuits Using Hot Luminescence. *IEEE Transactions on Electron Devices*, 18(7):330 –332, 1997.

[99] C. Kim and J.-J. Quisquater. Fault Attacks for CRT Based RSA: New Attacks, New Results, and New Countermeasures. In *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*, volume 4462 of *Lecture Notes in Computer Science*, pages 215–228. 2007.

[100] T. H. Kim, T. Takagi, D.-G. Han, H. W. Kim, and J. Lim. Side Channel Attacks and Countermeasures on Pairing Based Cryptosystems over Binary Fields. *IACR Cryptology ePrint Archive, Report 2006/243*, 2006.

[101] N. Koblitz. *Algebraic Aspects of Cryptography*. Algorithms and Computation in Mathematics. Springer, 1998.

[102] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO 1996, 16th Annual International Cryptology Conference*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[103] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO 1999, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[104] J. Krämer, M. Kasper, and J.-P. Seifert. The Role of Photons in Cryptanalysis. In *19th Asia and South Pacific Design Automation Conference, ASP-DAC 2014*, pages 780–787. IEEE, 2014.

[105] J. Krämer, D. Nedospasov, A. Schlösser, and J.-P. Seifert. Differential Photonic Emission Analysis. In E. Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - 4th International Workshop, COSADE 2013*, volume 7864 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2013.

[106] J. Krämer, D. Nedospasov, and J.-P. Seifert. Weaknesses in Current RSA Signature Schemes. In H. Kim, editor, *International Conference on Information Security and Cryptology - ICISC 2011*, volume 7259 of *Lecture Notes in Computer Science*, pages 155–168. Springer Berlin Heidelberg, 2012.

[107] J. Krämer, A. Stüber, and Ágnes Kiss. On the Optimality of Differential Fault Analyses on CLEFIA. *IACR Cryptology ePrint Archive, Report 2014/572*, 2014.

[108] M. Lanzoni, M. Manfredi, L. Selmi, E. Sangiorgi, R. Capelletti, and B. Ricco. Hot-Electron-Induced Photon Energies in n-Channel MOSFETs Operating at 77 and 300 K. *IEEE Transactions on Electron Devices*, 10(5):173–176, 1989.

[109] R. Lashermes, J. Fournier, and L. Goubin. Inverting the Final Exponentiation of Tate Pairings on Ordinary Elliptic Curves Using Faults. In G. Bertoni and J.-S. Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 365–382. Springer Berlin Heidelberg, 2013.

[110] R. Lashermes, M. Paindavoine, N. El Mrabet, J. J. A. Fournier, and L. Goubin. Practical Validation of Several Fault Attacks against the Miller Algorithm. In A. Tria and D. Choi, editors, *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2014, Busan, South Korea*, pages 115–122. IEEE Computer Society, 2014.

[111] V. Lomné, E. Prouff, M. Rivain, T. Roche, and A. Thillard. How to Estimate the Success Rate of Higher-Order Side-Channel Attacks. In L. Batina and M. Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2014.

[112] S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., 2007.

[113] S. Mangard, E. Oswald, and F.-X. Standaert. One for All - All for One: Unifying Standard Differential Power Analysis Attacks. *IET Information Security*, 5(2):100–110, 2011.

[114] M. Matsui and A. Yamagishi. A New Method for Known Plaintext Attack of FEAL Cipher. In R. A. Rueppel, editor, *Advances in Cryptology - EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 81–91, Berlin, Heidelberg, 1993. Springer-Verlag.

[115] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[116] A. Menezes, S. Vanstone, and T. Okamoto. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 80–89, 1991.

[117] T. S. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.

[118] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Examining Smart-Card Security under the Threat of Power Analysis Attacks. *IEEE Trans. Computers*, 51(5):541–552, 2002.

[119] B. Michéle, J. Krämer, and J.-P. Seifert. Structure-Based RSA Fault Attacks. In M. D. Ryan, B. Smyth, and G. Wang, editors, *Information Security Practice and Experience - 8th International Conference, ISPEC 2012*, volume 7232 of *Lecture Notes in Computer Science*, pages 301–318. Springer, 2012.

[120] V. S. Miller. Use of Elliptic Curves in Cryptography. In H. C. Williams, editor, *Advances in Cryptology - CRYPTO 1985, 5th Annual International Cryptology Conference*, volume 218 of *Lecture Notes in Computer Sciences*, pages 417–426. Springer-Verlag New York, Inc., 1986.

[121] V. S. Miller. The Weil Pairing, and Its Efficient Calculation. *J. Cryptology*, 17(4):235–261, 2004.

[122] J. A. Muir. Seifert's RSA Fault Attack: Simplified Analysis and Generalizations. In P. Ning, S. Qing, and N. Li, editors, *8th International Conference on Information and Communications Security, ICICS 2006*, volume 4307 of *Lecture Notes in Computer Science*, pages 420–434. Springer, 2006.

[123] D. Nedospasov, A. Schlösser, J.-P. Seifert, and S. Orlic. Functional Integrated Circuit Analysis. In *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*, 2012.

[124] D. Nedospasov and T. Schröder. Introducing Die Datenkrake: Programmable Logic for Hardware Security Analysis. In *7th USENIX Workshop on Offensive Technologies - WOOT 2013*. USENIX Association, 2013.

[125] R. Newman. Visible Light from a Silicon $p-n$ Junction. *Phys. Rev.*, 100:700–703, 1955.

[126] K. Nohl, D. Evans, Starbug, and H. Plötz. Reverse-Engineering a Cryptographic RFID Tag. In P. C. van Oorschot, editor, *Proceedings of the 17th USENIX Security Symposium, 2008, San Jose, CA, USA*, pages 185–194. USENIX Association, 2008.

[127] L. B. Oliveira, D. F. Aranha, C. P. Gouvêa, M. Scott, D. F. Câmara, J. López, and R. Dahab. TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks. *Computer Communications*, 34(3):485 – 493, 2011. Special Issue of Computer Communications on Information and Future Communication Security.

[128] L. B. Oliveira, D. F. Aranha, E. Morais, F. Daguano, J. López, and R. Dahab. TinyTate: Computing the Tate Pairing in Resource-Constrained Sensor Nodes. In *IEEE International Symposium on Network Computing and Applications (NCA 2007)*, pages 318–323. IEEE, 2007.

[129] L. B. Oliveira, D. F. Aranha, E. Morais, F. Daguano, J. López, and R. Dahab. TinyTate: Identity-Based Encryption for Sensor Networks. *IACR Cryptology ePrint Archive, Report 2007/020*, 2007.

[130] D. A. Osvik, A. Shamir, and E. Tromer. Cache Attacks and Countermeasures: The Case of AES. In D. Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006.

[131] C. Paar and J. Pelzl. *Understanding Cryptography. A Textbook for Students and Practitioners.* Springer-Verlag, 2010.

[132] D. Page and F. Vercauteren. Fault and Side-Channel Attacks on Pairing Based Cryptography. *IACR Cryptology ePrint Archive, Report 2004/283*, 2004.

[133] D. Page and F. Vercauteren. A Fault Attack on Pairing-Based Cryptography. *IEEE Transactions on Computers*, 55(9):1075–1080, 2006.

[134] M. Pavesi, P. Rigolli, M. Manfredi, P. Palestri, and L. Selmi. Spontaneous Hot-Carrier Photon Emission Rates in Silicon: Improved Modeling and Applications to Metal Oxide Semiconductor Devices. *Physical Review B*, 65(19):1–8, 2002.

[135] C. Percival. Cache Missing for Fun and Profit. In *Proceedings of BSDCan 2005*, 2005.

[136] G. Piret and J. Quisquater. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In C. D. Walter, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2003.

[137] E. Prouff, M. Rivain, and T. Roche. On the Practical Security of a Leakage Resilient Masking Scheme. *IACR Cryptology ePrint Archive, Report 2013/396*, 2013.

[138] J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In I. Attali and T. P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, 2001, Proceedings*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[139] J. M. Rabaey and A. Chandrakasan. *Digital Integrated Circuits*. A Design Prespective. Pearson Education, second edition, 2003.

[140] W. Rankl and W. Effing. *Smart Card Handbook*. Wiley, fourth edition, 2010.

[141] C. Rebeiro, R. Poddar, A. Datta, and D. Mukhopadhyay. An Enhanced Differential Cache Attack on CLEFIA for Large Cache Lines. In D. J. Bernstein and S. Chatterjee, editors, *INDOCRYPT 2011 - 12th International Conference on Cryptology in India*, volume 7107 of *Lecture Notes in Computer Science*, pages 58–75. Springer, 2011.

[142] C. Rebeiro, A. D. Selvakumar, and A. S. L. Devi. Bitslice Implementation of AES. In D. Pointcheval, Y. Mu, and K. Chen, editors, *5th International Conference on Cryptology and Network Security, CANS 2006, Suzhou, China, 2006, Proceedings*, volume 4301 of *Lecture Notes in Computer Science*, pages 203–212. Springer, 2006.

[143] M. Rivain. Securing RSA against Fault Analysis by Double Addition Chain Exponentiation. In M. Fischlin, editor, *Topics in Cryptology - CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 459–480. Springer, 2009.

[144] A. Sahai and B. Waters. Fuzzy Identity-Based Encryption. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer Berlin Heidelberg, 2005.

[145] K. Sakiyama, Y. Li, M. Iwamoto, and K. Ohta. Information-Theoretic Approach to Optimal Differential Fault Analysis. *IEEE Transactions on Information Forensics and Security*, 7(1):109–120, 2012.

[146] W. Schindler. A Timing Attack against RSA with the Chinese Remainder Theorem. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 109–124. Springer, 2000.

[147] W. Schindler. Advanced Stochastic Methods in Side Channel Analysis on Block Ciphers in the Presence of Masking. *J. Mathematical Cryptology*, 2(3):291–310, 2008.

[148] W. Schindler, K. Lemke, and C. Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.

[149] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert. Simple Photonic Emission Analysis of AES. *J. Cryptographic Engineering*, 3(1):3–15, 2013.

[150] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert. Simple Photonic Emission Analysis of AES - Photonic Side Channel Analysis for the Rest of Us. In E. Prouff and P. Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 41–57. Springer, 2012.

[151] B. Schneier. Another New AES Attack. https://www.schneier.com/blog/archives/2009/07/another_new_aes.html, 2009. [accessed 2014/10/15].

[152] M. Scott. Computing the Tate Pairing. In A. Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 293–304. Springer, 2005.

[153] M. Scott. On the Efficient Implementation of Pairing-Based Protocols. In L. Chen, editor, *IMA Int. Conf.*, volume 7089 of *Lecture Notes in Computer Science*, pages 296–308. Springer Berlin Heidelberg, 2011.

[154] M. Scott, N. Benger, M. Charlemagne, L. J. D. Perez, and E. J. Kachisa. On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves. In H. Shacham and B. Waters, editors, *Pairing 2009, Palo Alto, CA, USA, 2009, Proceedings*, volume 5671 of *Lecture Notes in Computer Science*, pages 78–88. Springer, 2009.

[155] J.-P. Seifert. On Authenticated Computing and RSA-Based Authentication. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 122–127. ACM, 2005.

[156] L. Selmi, M. Mastrapasqua, D. Boulin, J. Bude, M. Pavesi, E. Sangiorgi, and M. Pinto. Verification of Electron Distributions in Silicon by Means of Hot Carrier Luminescence Measurements. *IEEE Transactions on Electron Devices*, 45(4):802 –808, 1998.

[157] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[158] P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.*, 26(5):1484–1509, Oct. 1997.

[159] J. H. Silverman. *The Arithmetic of Elliptic Curves. 2nd ed.* Graduate Texts in Mathematics 106. New York, NY: Springer Berlin Heidelberg, 2009.

[160] S. Skorobogatov. Using Optical Emission Analysis for Estimating Contribution to Power Analysis. In L. Breveglieri, I. Koren, D. Naccache, E. Oswald, and J.-P. Seifert, editors, *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 111 –119. IEEE Computer Society, 2009.

[161] P. Song, F. Stellari, B. Huott, O. Wagner, U. Srinivasan, Y. Chan, R. Rizzolo, H. Nam, J. Eckhardt, T. McNamara, C.-L. Tong, A. Weger, and M. McManus. An Advanced Optical Diagnostic Technique of IBM z990 eServer Microprocessor. *IEEE International Test Conference, 2005*, pages 1227 – 1235, 2005.

[162] F.-X. Standaert, B. Gierlichs, and I. Verbauwhede. Partition vs. Comparison Side-Channel Distinguishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices. In *International Conference on Information Security and Cryptology - ICISC 2008*, pages 253–267, 2008.

[163] W. Stein et al. Sage Mathematics Software (Version 6.1). http://www.sagemath.org. [accessed 2014/12/15].

[164] S. Sze and K. Ng. *Physics of Semiconductor Devices.* Wiley, 2006.

[165] M. M. I. Taha and P. Schaumont. A Novel Profiled Side-Channel Attack in Presence of High Algorithmic Noise. In *30th International IEEE Conference on Computer Design, ICCD 2012*, pages 433–438. IEEE Computer Society, 2012.

[166] S. Tajik, E. Dietz, S. Frohmann, J.-P. Seifert, D. Nedospasov, C. Helfmeier, C. Boit, and H. Dittrich. Physical Characterization of Arbiter PUFs. In L. Batina and M. Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 493–509. Springer, 2014.

[167] S. Tam, F. Hsu, P. Ko, C. Hu, and R. Muller. Spatially Resolved Observation of Visible-Light Emission from Si MOSFET's. *IEEE Transactions on Electron Devices*, 4(10):386–388, 1983.

[168] A. Toriumi, M. Yoshimi, M. Iwase, Y. Akiyama, and K. Taniguchi. A Study of Photon Emission from n-Channel MOSFET's. *IEEE Transactions on Electron Devices*, 34(7):1501–1508, 1987.

[169] A. Tosi, F. Stellari, A. Pigozzi, G. Marchesi, and F. Zappa. Hot-Carrier Photoemission in Scaled CMOS Technologies: A Challenge for Emission Based Testing and Diagnostics. In *Reliability Physics, 2006*, pages 595–601, 2006.

[170] E. Trichina and R. Korkikyan. Multi Fault Laser Attacks on Protected CRT-RSA. In L. Breveglieri, M. Joye, I. Koren, D. Naccache, and I. Verbauwhede, editors, *2010 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 75–86. IEEE Computer Society, 2010.

[171] J. C. Tsang and M. V. Fischetti. Why Hot Carrier Emission Based Timing Probes Will Work for 50 nm, 1V CMOS Technologies. *Microelectronics Reliability*, pages 1465–1470, 2001.

[172] J. C. Tsang, J. A. Kash, and D. P. Vallett. Picosecond Imaging Circuit Analysis. *IBM Journal of Research and Development*, 44(4):583 –603, 2000.

[173] Télécom ParisTech. DPA Contest v4. http://www.dpacontest.org. [accessed 2014/11/14].

[174] I. Verbauwhede, D. Karaklajic, and J.-M. Schmidt. The Fault Attack Jungle - A Classification Model to Guide You. In L. Breveglieri, S. Guilley, I. Koren, D. Naccache, and J. Takahashi, editors, *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 3–8. IEEE, 2011.

[175] F. Vercauteren. The Hidden Root Problem. In S. D. Galbraith and K. G. Paterson, editors, *Pairing-Based Cryptography - Pairing 2008, Second International Conference, Egham, UK, 2008. Proceedings*, volume 5209 of *Lecture Notes in Computer Science*, pages 89–99. Springer, 2008.

[176] F. Vercauteren. Pairings on Elliptic Curves. In M. Joye and G. Neven, editors, *Identity-Based Cryptography*, volume 2 of *Cryptology and Information Security Series*, pages 13–30. IOS Press, 2009.

[177] E. R. Verheul. Evidence that XTR Is More Secure than Supersingular Elliptic Curve Cryptosystems. In B. Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 195–210. Springer Berlin Heidelberg, 2001.

[178] S. Villa, A. Lacaita, and A. Pacelli. Photon Emission From Hot Electrons in Silicon. *Physical Review B*, 52(15):10993–10999, 1995.

[179] N. H. E. Weste and D. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison Wesley, fourth edition, 2010.

[180] C. Whelan and M. Scott. The Importance of the Final Exponentiation in Pairings When Considering Fault Attacks. In T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors, *Pairing*, volume 4575 of *Lecture Notes in Computer Science*, pages 225–246. Springer Berlin Heidelberg, 2007.

[181] C. Whitnall, E. Oswald, and F.-X. Standaert. The Myth of Generic DPA... and the Magic of Learning. In J. Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 183–205. Springer International Publishing, 2014.

[182] P. Wright. *Spycatcher: The Candid Autobiography of a Senior Intelligence Officer*. Viking, 1987.

[183] Z.-F. Zhang, J. Xu, and D.-G. Feng. Attack on an Identification Scheme Based on Gap Diffie-Hellman Problem. *IACR Cryptology ePrint Archive, Report 2003/153*, 2003.