Dissertation

Physical attacks on pairing-based cryptography

Peter Günther

May 31, 2016 (revised version)

Submitted to the

Department of Computer Science
Paderborn University

in partial fulfillment of the requirements for the degree of

Doktor der Naturwissenschaften
(doctor rerum naturalium)

Supervisor: Prof. Dr. rer. nat. Johannes Blömer

Commission:

Prof. Dr. Johannes Blömer (first reviewer)
Dr. Paul Kaufmann
Prof. Dr. Friedhelm Meyer auf der Heide
Prof. Dr. Marco Platzner
Prof. Dr. Jean-Pierre Seifert (second reviewer)

for Katrin, Mathilda, and Anton

In this thesis, we analyze the vulnerability of pairing-based cryptographic schemes against physical attacks like side-channel attacks (SCAs) or fault attacks (FAs). Compared to well-established cryptographic schemes, for example, from standard elliptic curve cryptography (ECC), less is known about weaknesses of pairing-based cryptography (PBC) against those attacks. Reasons for this shortcoming are the complexity of PBC and a missing consensus on parameters, algorithms, and schemes, e.g., in the form of standards. Furthermore, the structural difference between ECC and PBC permits a direct application of the results from ECC. To get a better understanding of the subject, we present new physical attacks on PBC.

In an attack on pairing-based encryption schemes, one argument of the pairing is secret and the other argument is known to the attacker. With respect to SCAs, it seems easier to attack the pairing computation in the case where the secret is the second argument of the pairing. So far, it was not completely clear how much protection it offers to use the secret as the first argument of the pairing. To show that both choices are equally vulnerable, we present new attacks for the case where the secret is the first argument of the pairing.

With respect to FAs on the pairing computation, we introduce a framework for the analysis of erroneous computations that covers all techniques from previous attacks. Based on this framework, we describe two new attacks. The novelty of the attacks is that they consider both steps of the pairing computation, the Miller algorithm and the final exponentiation. Furthermore, to the best of our knowledge, we provide the first practical realization of an FA on pairings that includes both steps of the computation.

So far, all previous investigations concentrate on the pairing computation itself. But elliptic curve scalar multiplication (ECSM) with a secret scalar is also used in PBC. In several pairing-based signature schemes, the base point of the ECSM is the image of a hash function that maps strings to the elliptic curve. The computation of the hash function uses point decompression as a bulding block. We present an FA on point decompression such that points are decompressed to a singular curve where the discrete logarithm problem is trivial. This allows us to compute the secret scalar and hence, to break the corresponding signature schemes. Consequently, we provide a practical realization of our attack for the pairing-based short signature scheme of Boneh, Lynn, and Shacham.

Our results, including the practical realizations of our attacks, show that physical attacks are a threat for PBC and need further investigation. Our work also shows that the community should agree on parameters, algorithms, and schemes to reduce the complexity of PBC with respect to physical attacks.

Acknowledgments

First of all, I thank Johannes Blömer that he gave me the opportunity to work on the subject of this thesis, even though I had only little knowledge about cryptography when I started back in 2010. Beyond the constant feedback on my work, I thank Johannes for the scientific education, a generous traveling budget, last-minute reviews of my introductions, and for the help to find my own way without getting lost.

Furthermore, I thank Jean-Pierre Seifert for the feedback on this thesis and on my work in general. I also thank Jean-Pierre for his hospitality during my visits to TU Berlin and for the access to his Lab.

I thank my co-authors Juliane Krämer, Ricardo Gomes da Silva, Volker Krummel, and Gennadij Liske for the exciting cooperation, for blasting UPB's telephone bills, and for sharing conference hotel rooms. I also thank them for the thorough proof-reading.

Of course, I also thank my (former) colleagues at Uni Paderborn for the good times. In particular, I thank Kathrin Bujna, Stefanie Mense, Marcel Ackermann, Jan Bobolz, Sascha Brauer, Fabian Eidens, Jakob Juhnke, Daniel Kuntze, Gennadij Liske, Nils Löken, Ronald Petrlic, Christoph Sorge, and David Teusner for feedback, discussions, and proof-reading. Thanks to Uli Ahlers and Heinz-Georg Wassing for technical support, and to Thomas Thissen for taking the picture of Figure 8.2. I thank Claudia Jahn for encouraging me to leave Hamburg for Paderborn in order to start the job here.

Furthermore I thank Dmitry Nedospasov for his advice on fault attacks and Sen Wu for his support with measurements, in particular for Figure 1.2. I thank Michelle Kloppenburg and Christian Günther for proof-reading.

Finally, I thank my family, especially Katrin, Mathilda, Anton, and my parents(-in-law) for their patience, support, and for keeping themselves busy with each other while I was busy with the thesis.

I was only able to complete this thesis with the support from the German Federal Ministry of Education and Research within the projects "Systemintegrität für Selbstbedienungssysteme" (grant 01IS10030C) and "Securing the Financial Cloud" (grant 16KIS0062).

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1.

Introduction

In this thesis, we study attacks on cryptographic implementations that exploit physical properties during the execution of an algorithm. More specifically, we are interested in attacks on cryptographic schemes that use bilinear pairings on elliptic curves as a building block.

1.1. The story of Alice and Bob

Let us consider the following scenario: Alice and Bob live in different corners of the world. Bob likes to send his private news to Alice via the Internet and, in times of online dating, he likes to do this even before he has met Alice personally for the first time. Bob could post his message on the social medias in the clear, like everybody does today. Maybe Bob is old-fashioned, but he prefers to communicate confidentially. Furthermore, he does not trust the Internet because Alice's big sister Eve eavesdrops all communication in and out of the house.

What can Alice and Bob do? In 1976, Diffie and Hellman [DH76] introduced the concept of *asymmetric* encryption that partly solves their issue. In an asymmetric encryption scheme, Alice as the recipient of a message, has a so-called public-private key pair. We can think of this pair as a snap-lock with a matching key. Here, the snap-lock represents the public key and the key represents the private key. The public key is used to encrypt, respectively lock, messages. The private key is used to decrypt, respectively unlock, messages. With asymmetric encryption, the story of Alice and Bob goes like this:

1. First, Alice executes a key generation algorithm to generate a secret private key sk and a corresponding public key pk.
2. She publishes her public key pk in a public key directory on the Internet.
3. Bob downloads Alice's public key pk to encrypt his message M for Alice with this key. He obtains a ciphertext C that he sends to Alice over the Internet.
4. Alice retrieves the ciphertext C and with her matching secret key sk she is able to recover the message from the ciphertext.

For a secure asymmetric encryption scheme, of course, nobody is able to compute the secret key from the public key; and nobody is able to decrypt the ciphertext without knowledge of the secret key.

The first published instantiation of an asymmetric encryption scheme was presented in 1978 by Rivest, Shamir, and Adleman [RSA78] and named after the authors: the Rivest-Shamir-Adleman (RSA) cryptosystem. Many instantiations of asymmetric encryption are known today and used by all of us, for example, in Transport Layer Security (TLS).

Unfortunately, asymmetric encryption still does not solve all the problems of Alice and Bob. How can Bob be sure that he really uses the public key of Alice for encrypting his secret message? Suppose nosy Eve publishes her own public key, but in the name of Alice. Then Bob might encrypt his message for Alice with Eve's key. Eve eavesdrops the ciphertext and decrypts it with her private key. The standard approach for solving this problem are certificates. Alice, before publishing her public key, asks a trusted certification authority to issue a signed certificate that confirms that this key belongs to the identity Alice. Then Bob, before encrypting his message for Alice, checks that the key he downloaded is really certified for Alice. In a huge infrastructure like the Internet (of Things), this results in a complicated *public key infrastructure*.

A solution to get rid of the multitude of authenticated public keys is identity-based encryption (IBE). In IBE, the public encryption key of Alice does not depend on her secret decryption key anymore. Instead, Bob uses a unique string that encodes Alice's identity, for example, her email address `alice@home.de`. Furthermore, the trusted certification authority is replaced with a trusted private key center that issues private keys for users like Alice. Then Bob encrypts his message with a combination of Alice's identity `alice@home.de` and the public key of the private key center. Hence, Bob has to retrieve only the authenticated public key of the private key center to encrypt messages for arbitrary identities. This results in a simplified public key infrastructure, especially in large-scale networks with authenticated identities. The concept of IBE was already introduced in 1985 by Shamir [Sha85]. But the first fully functional scheme was given by Boneh and Franklin [BF01] in 2001. The construction is based on pairings, like most of the efficient constructions of IBE and more powerful primitives like attribute-based encryption (ABE).

We remark that today, confidential communication is not the only objective of cryptography. For example, we like to authenticate our communication, we like to sign contracts over the Internet, we like to pay anonymously in the Internet, and we like to play mental poker over the telephone. Research has worked hard on all of these applications and came up with solutions for many more of them. For example, digital signature schemes were already defined by Diffie and Hellman [DH76]. Digital signatures are *publicly verifiable* and provide *authenticity*, *integrity*, and *non-repudiation*. Hence, with respect to authenticity and non-repudiation, they are the counterpart of classical pen and paper signatures and, additionally, provide integrity of the message.

1.2. Modern cryptography: from fuzzy to fussy

Modern cryptography started in the second half of the 20^{th} century. The central aspects of modern cryptography are: "emphasis on definitions, precise assumptions, and rigorous proofs of security" [KL08]. Hence, for any cryptographic primitive, the first step is to formally define its syntax, its functionality, and its security. Another important paradigm of modern cryptography is Kerckhoffs' principle that requires that security solely relies on the secrecy of keys and not on the secrecy of algorithms.

1.2.1. Asymmetric encryption

To get an impression of what modern cryptography looks like, we come back to asymmetric encryption. Regarding the *syntax*, an asymmetric encryption scheme consists of three polynomial time algorithms:

1. A key generation algorithm that generates a public-private key pair.
2. An encryption algorithm that takes as input a public key and a message and outputs a ciphertext.
3. A decryption algorithm that takes as input a private key and a ciphertext and outputs a message.

Assume that a ciphertext was computed by the encryption algorithm for a given message and a given public key. The *functionality* of an asymmetric encryption scheme requires that the decryption algorithm recovers the original message when it is executed on this ciphertext and the private key that corresponds to the public key that was used for encryption.

The definition of *security* is more involved. Informally, we like to say that an encryption scheme is secure if no polynomial time adversary can gain any information about the message of a given ciphertext without knowledge of the corresponding secret key. Furthermore, this should still hold true, even if the adversary is able to query the decryption of additional ciphertexts for the same target secret key. This informal description of security can be formalized and is called *indistinguishability* against *chosen ciphertext attacks*, or IND-CCA security. Today, this is accepted as the right notion of security for encryption schemes. For the formal definition we refer, e.g., to Definition 10.24 of [KL08].

Based on the definitions for syntax, functionality, and security of a cryptographic primitive, schemes for this primitive can be described. Then it is proven, relative to clearly stated hardness assumptions, that a particular scheme satisfies the corresponding definitions.

1.2.2. The discrete logarithm problem

We now introduce a concrete hardness assumption that plays an important role in this thesis: the discrete logarithm (DLOG) problem. Let $\mathcal{G}$ be a cyclic group of

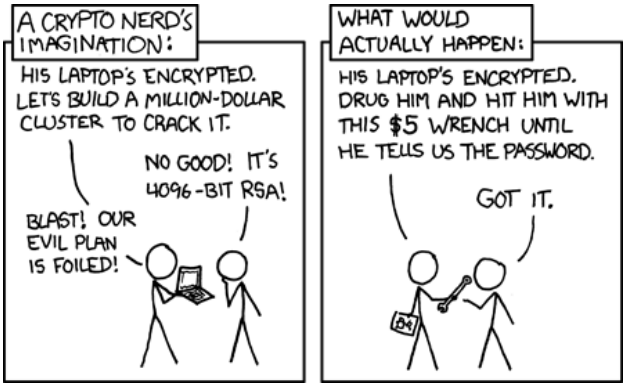

Figure 1.1.: Black-box attacks versus physical attacks (`http://xkcd.com/538/`): Often secure schemes can be broken at much lower costs by exploiting physical properties of the implementation.

order n with generator g and let $h \in \mathcal{G}$. Given $(\mathcal{G}, n, g, h)$, the *discrete logarithm (DLOG) problem* is to compute an integer α such that $g^\alpha = h$. Here, α is called the DLOG of h to the basis g. Informally, the DLOG problem is hard in a group $\mathcal{G}$ if no polynomial time algorithm can solve the DLOG problem with significant success probability over the random choices of h.

One possible candidate for a family of groups where the DLOG problem is hard are elliptic curves. An elliptic curve is a curve defined by a cubic equation in x and y of the form

$$y^2 = x^3 + a_4 x + a_6.$$

On the set of points $P = (x, y)$ that satisfy this equation, we can define an abelian group. Traditionally, this group is written as an additive group. For a point P on the curve and a positive integer α we write αP to denote the sum $P + P + \cdots + P$, α times. We call the computation of αP from α and P elliptic curve scalar multiplication (ECSM). Hence, the DLOG problem is to compute α from P and αP, i.e., to invert ECSM. Elliptic curve cryptography (ECC) is based on the assumption that the DLOG problem on elliptic curves is hard. The reason to believe in this assumption is that, for suitable elliptic curves, the best known algorithms to solve the DLOG problem with significant probability have exponential run time. And indeed, the security of wide-spread schemes like the elliptic curve integrated encryption scheme (ECIES) and elliptic curve digital signature algorithm (ECDSA) relies on the hardness of the DLOG problem on elliptic curves.

1.3. The real world: getting physical

The standard security definitions like IND-CCA security are based on black-box models where we assume that an adversary is only able to observe or manipulate the

```
h ← g
for α_i ← α_{t−2} ... α_0 do
    h ← h^2
    if α_i equals 1 then
        h ← hg
    end if
end for
```

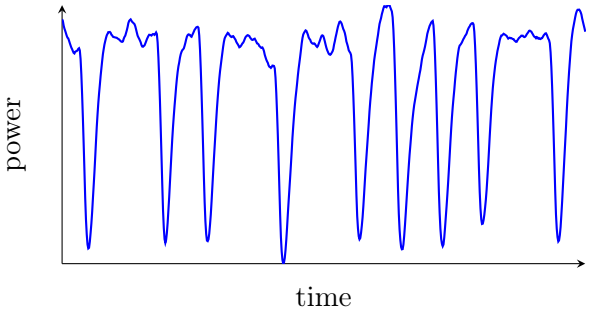

Figure 1.2.: On the left, we see the standard square and multiply algorithm for computing $h = g^\alpha$ for a t-bit integer α with binary representation $\alpha_{t-1} \dots \alpha_0$. On the right, we see the low-pass filtered power consumption of an AVR ATmega163 during the computation. More specifically, we see the processing of one byte 10110001 (binary) of α. It is clearly possible to distinguish bits of α that are set to 1 (wide pulse) from bits that are set to 0 (narrow pulse).

public input and output of cryptographic algorithms. But in practice, implementations of these algorithms are executed on real hardware and the involved secret keys are physically stored. Therefore, in practice, often more powerful adversaries occur that do not adhere to the black-box model and that exploit physical properties. Basically, there are two types of physical attacks:

Side-channel attacks (SCAs) In a (passive) side-channel attack (SCA), the adversary observes different physical properties during the execution of a cryptographic algorithm such as power consumption, timing, or electromagnetic (EM) radiance.[1]

Fault attacks (FAs) In an (active) fault attack (FA), the adversary tampers with the executing device prior to or during the execution of a cryptographic algorithm, for example, by means of electromagnetic pulses, nuclear radiation, or laser beams.

Both types of attacks might give the adversary auxiliary information about intermediate states of the algorithm, additional to the information he gains from the output of the algorithm in a black-box attack. This additional information might enable the attacker to break the system or to reduce the costs of an attack (cf. Figure 1.1).

In Figure 1.2 we see an example for an SCA based on the power consumption of a cryptographic device that computes $h = g^\alpha$ from the most significant bit (MSB) to the least significant bit (LSB). For the bits of α that are set to 1, a squaring is followed by a multiplication and a wide pulse occurs. For the bits that are set to 0,

[1] The term SCA is sometimes used to refer to both, passive as well as active physical attacks. As, e.g., [MOP07] we use SCA only to refer to passive physical attacks.

only squaring is computed and hence, we observe a narrow pulse. This allows us to directly read off α from the power profile of the computation.

One early example of an SCA is described by the former British intelligence officer Peter Wright in [Wri87]: In 1956, during the Suez Crisis, the British broke the Egyptian Hagelin mechanical cipher machine with an acoustic SCA to eavesdrop the communication between Egypt and the Soviet Union. In academia, one of the first SCAs was described in 1996 by Kocher in [Koc96]. Boneh, DeMillo, and Lipton [BDL97] described the first FA in 1997 that targets an implementation of RSA. This attack, also known as the Bellcore attack, was shown to be practical in [Aum+03]. Since then, numerous new attacks have been presented every year [Wag12].

With respect to SCAs on modern ciphers, the first attacks were based on information leakage through timing [Koc96] and power consumption [KJJ99]. Attacks based on electromagnetic radiance followed [QS01; GMO01]. More and more physical side-channels were exploited, for example photonic emission [FH08; Sch+13], and again, acoustic noise [GST14]. The techniques to access this information also improved: Starting with reading of bits directly from the power consumption of a device in simple power analysis, today, powerful tools that use statistics like differential power analysis (DPA) [KJJ99] exist.

For FAs, more and more different fault injection techniques were developed over time. Examples include clock and power glitches [AK96], laser beams [SA02], or EM disturbance [HS07]. Furthermore, different techniques to exploit these faults were proposed such as differential fault analysis [BS97; BMM00], safe-error analysis [YJ00], weak-curve based analysis [BMM00], and sign-change attacks [BOS06]. Today also higher order attacks are practical. These are attacks where multiple side-channels are combined [Mes00] or multiple faults are introduced within one execution [KQ07].

Even though also examples exist where SCAs as well as FAs are carried out remotely [BB05; GMM15], typically, the adversary requires physical access to the device that contains the target key. One may wonder why we consider attacks where an adversary already has physical access to the device that stores the target key. The reason is that in many applications, the secret key is stored on special tamper-protected hardware, like a smart card, such that not even the legitimate owner of the key can access it directly. This prevents cloning of keys or protection in case of theft. For example, in a pay TV application a regular subscriber should not be able to clone and distribute the decryption key. Another example is electronic signatures according to the German Digital Signature Act [BRD01]. To issue qualified signatures, the signing key has to be stored on a device like a smart card that prevents cloning of the key. We see that cryptographic implementations on smart cards are a profitable target of physical attacks.

1.4. Pairing-based cryptography

In pairing-based cryptography (PBC) the basic building block is a cryptographic pairing, i.e., a map

$$\mathrm{e} : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$$

for groups $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_T$. Throughout this thesis, as for all cryptographic pairings used today, $\mathcal{G}_1$ and $\mathcal{G}_2$ are subgroups of an elliptic curve that is defined over a finite field. Furthermore, $\mathcal{G}_T$ is a subgroup of the field's multiplicative group. Usually, we assume that a pairing satisfies two functional properties: bilinearity, i.e., linearity in both arguments, and efficiency. To be useful in cryptography, also some security properties are required, e.g., that it is hard to invert the pairing for both of its arguments.

In cryptography, the first application of pairings was destructive. Menezes, Okamoto, and Vanstone [MOV93] used pairings to reduce the DLOG problem on certain elliptic curves to the DLOG problem in a finite field. Hence, they showed that the DLOG problem on some elliptic curves, especially so-called supersingular curves, has sub-exponential complexity. In 2001, Boneh and Franklin gave a constructive application and instantiated an efficient IBE scheme based on pairings. Today numerous schemes other than IBE use pairings as their building blocks. Examples include attribute-based encryption [SW05], key agreement [Jou04], non-interactive zero-knowledge proofs [GOS06], short signatures [BLS04b], group signatures [BS04], or threshold and blind signatures [Bol03].

The adoption of pairings in cryptographic applications is followed by the request for efficient implementations. Over the past years, research efforts led to pairings that have efficient implementations and can be implemented on resource constrained devices such as smart cards [SCA06; GK15]. This clears the way for the application of PBC in adversarial environments where physical attacks are relevant.

Starting in 2001, PBC is a relatively young field of research. And although PBC uses methods from ECC, the vulnerability of PBC against physical attacks is not well understood. In nearly all ECC-based schemes, the secret is a scalar multiplier of a point on the curve. In PBC, the secret plays different roles. It can either be a scalar like in ECC [BLS04b] or it is a point on the curve [BF01; CC03]. This point can be a generator and input to elliptic curve scalar multiplication [CC03]. But it can also be an argument of the pairing [BF01]. Furthermore, it depends on the concrete application whether it is the first argument or the second argument of the pairing. Finally, various pairings such as Weil pairing, Tate pairing, Ate pairing, and Eta pairing exist. The situation appears much more complex than for RSA or ECC where the secret is simply a scalar that is the input of an exponentiation or a scalar multiplication.

1.5. Thesis statement and contribution

On the one hand, in the black-box model, the community agreed on models for the security of many cryptographic primitives. On the other hand, many years after the first publications of physical attacks, there is still no agreement on how to model physical adversaries. Attempts to formalize SCAs like [MR04; DP08; SMY09; DDF14] often result in impractical constructions and still they do not capture all practical attacks. Hence, there is an ongoing dispute between theory and practice [KM13]. In practice, it is therefore still inevitable to design countermeasures that are specialized on one or several types of physical attacks. Then, an implementation that is secure in the black-box model is hardened with these countermeasures to resist the designated attacks. For this approach, an important first step is to understand vulnerabilities of cryptographic schemes to physical attacks. Although the first attacks on RSA were already published in 1996, this analysis is still ongoing and possibilities for new attacks are discovered regularly. The same holds for ECC.

Despite the complexity of PBC, the effort that has been spent on the analysis of physical attacks is much smaller for PBC than for RSA or standard ECC. Even though there are some results that analyze the vulnerability of pairings to passive attacks [PV04; WS06; Kim+06; MFN09; GC11; UW14] as well as to active attacks [PV06; WS07; Mra09; BMH13; Las+14; LFG13] there are still some open questions.

> *The goal of this thesis is to analyze PBC with respect to physical attacks and to discover new vulnerabilities. With this contribution, we would like to prevent mistakes in future implementations. Furthermore, our results provide input for the design of new countermeasures and for risk assessment.*

More specifically, in this thesis we answer several questions that were raised in the context of physical attacks on PBC:

1. The pairing computation is not symmetric in its arguments. Whelan and Scott [WS06] investigate the question if it is more secure in the context of SCAs, and especially a DPA, to use the secret as the first or as the second argument of the pairing. They conclude that using the secret as the first argument might be more secure, especially when finite field multiplications are targeted in the SCA. This is not generally the case because it was already shown that the pairing is vulnerable to a DPA of finite field additions, also in the case where the first argument is secret [MFN09; GC11]. In Chapter 4 we complement this work and show that the pairing computation with secret first argument is similarly vulnerable to a DPA of finite field multiplications.

2. Most pairings are computed in two steps: Miller algorithm and final exponentiation. Whelan and Scott [WS07] conjecture that this two-step approach offers some protection against FAs on the pairing computation. Furthermore, several attacks on the pairing computation analyze either the Miller algorithm

[Mra09; BMH13; Las+14] *or* the final exponentiation [LFG13], but ignore the other step. In Chapter 5 we develop a framework for the analysis of FAs on pairings that combines the analysis of both steps. Furthermore, in Chapter 6 we present new second order attacks in the context of our framework that consider the Miller algorithm *and* the final exponentiation.

3. Previous investigations of physical attacks against PBC restrict to the pairing computation itself. But in PBC also other building blocks are combined in a different way from standard ECC. For example, in various pairing-based signature schemes, strings are hashed to a point on an elliptic curve and the result is used as input for ECSM with a secret scalar. The question is if this offers new vulnerabilities to physical attacks. In Chapter 7 we show that this is the case by presenting a new FA that applies to several pairing-based signature schemes.

4. So far, most FAs on PBC were described only theoretically. Especially it was an open question if second order attacks on the complete pairing computation, including Miller algorithm and final exponentiation, are practical. In Chapter 8, we show that this is the case. Therefore, we present the practical realization of one of our second order FAs on pairings from Chapter 6. Furthermore, we also present the practical realization of our attack against pairing-based signature schemes from Chapter 7.

1.6. Publications related to this thesis

This thesis is based on the following publications with substantial contribution of the author:

[BG15] Johannes Blömer and Peter Günther. "Singular Curve Point Decompression Attack." In: *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2015*. Ed. by Naofumi Homma and Victor Lomné. IEEE Computer Society, 2015, pp. 71–84.

[BGL13] Johannes Blömer, Peter Günther, and Gennadij Liske. "Improved Side Channel Attacks on Pairing Based Cryptography." In: *Proceedings of Constructive Side-Channel Analysis and Secure Design (COSADE) 2013*. Ed. by Emmanuel Prouff. Vol. 7864. LNCS. Springer, 2013, pp. 154–168.

[Blö+14] Johannes Blömer, Ricardo Gomes da Silva, Peter Günther, Juliane Krämer, and Jean-Pierre Seifert. "A Practical Second-Order Fault Attack against a Real-World Pairing Implementation." In: *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2014*. Ed. by Assia Tria and Dooho Choi. IEEE Computer Society, 2014, pp. 123–136.

Furthermore, the author of this thesis also contributed to the following publications that are related to PBC:

[BGK13] Johannes Blömer, Peter Günther, and Volker Krummel. "Securing Critical Unattended Systems with Identity Based Cryptography — A Case Study." In: *Proceedings of Mathematical Aspects of Computer and Information Sciences (MACIS) 2013*. 2013, pp. 98–105.

[BGL14] Johannes Blömer, Peter Günther, and Gennadij Liske. "Tampering Attacks in Pairing-Based Cryptography." In: *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2014*. Ed. by Assia Tria and Dooho Choi. IEEE Computer Society, 2014, pp. 1–7.

[GK15] Peter Günther and Volker Krummel. "Implementing Cryptographic Pairings on Accumulator based Smart Card Architectures." In: *Proceedings of Mathematical Aspects of Computer and Information Sciences (MACIS) 2015*. Ed. by Ilias S. Kotsireas, Siegfried M. Rump, and Chee K. Yap. Vol. 9582. LNCS. Springer, 2015, pp. 151–165.

1.7. Organization of this thesis

This thesis is organized as follows:

Chapter 2 In this chapter, we define the notation and give the required mathematical background, especially on elliptic curves and cryptographic pairings. We collect several theorems with relevance for PBC that are otherwise scattered over the literature. Because we provide cross references in later chapters, the reader may feel free to skip this chapter.

Chapter 3 In this chapter, we give background information on implementations and applications of pairings in cryptography. Our outline exceeds the pure mathematical definitions and provides information on terminology and typical implementations of ECC and PBC as far as required to understand this thesis. Furthermore, it presents examples for different types of cryptographic schemes from ECC and PBC that are relevant for this thesis.

This chapter is meant to understand the setting of our attacks. But together with Section 2.2 and Section 2.3 it provides a brief introduction into PBC, especially for the novice in the field. The reader that is familiar with PBC may skip this part because we later on refer to the corresponding definitions. Nevertheless, to get a quick idea of what kind of schemes we target with our attacks, we suggest Section 3.3.

Chapter 4 This chapter presents the first new results of this thesis. Based on our work from [BGL13; BGL11], we present passive SCAs on pairing-based encryption schemes. We show that unprotected implementations of pairings are vulnerable to a DPA of finite field multiplications. We do this by identifying

multiplications in the pairing computation that are suitable for a DPA and that allow recovery of the secret key. We present three attacks that all exploit common efficient representations of points on an elliptic curve.

Chapter 5 In this chapter, we introduce a framework for the analysis of FAs on pairings. It summarizes the existing techniques and presents them in a consistent way. Initially, our motivation was to provide a first step towards the automated analysis of FAs on pairings. But the framework also provides a better understanding of potential vulnerabilities of pairing computations against FAs. We apply the framework to some existing attacks from the literature.

Chapter 6 This chapter is based on our results from [Blö+14]. We present two new FAs on pairings that we analyze based on the framework from Chapter 5. Previous attacks target either the Miller loop or the final exponentiation, whereas for a complete attack on a relevant pairing, both steps have to be considered. We present an attack in the instruction skip model that includes both steps of the pairing computation. In [Blö+14] we presented an attack that completely removes the final exponentiation. For some implementations, this is not possible. In this chapter, we extend the results of [Blö+14] and present a new technique to attack the final exponentiation without completely removing it. Finally, we apply this technique to the combination of the optimal Ate pairing with Barreto-Naehrig (BN) curves [Ver10].

Chapter 7 In this chapter, we present our results from [BG15] where we introduced a new weak curve attack on ECC. We show that in the instruction skip model, it is possible to mount a FA such that compressed points are decompressed to a singular curve. Singular curves are weak and this allows us to solve the DLOG on the singular curve. Our attack has applications in point decompression, hashing to elliptic curves, and random point sampling. Furthermore, it can be mounted on real schemes. Most vulnerable are a class of pairing-based signature schemes that hash strings to elliptic curves like, for example, the Boneh-Lynn-Shacham (BLS) short signature scheme.

Chapter 8 In this chapter, we give detailed background information on the practical realization of some of our attacks from Chapter 6 and Chapter 7. We describe a setup to implement instruction skips by means of clock glitching. Furthermore, we describe a strategy to identify parameters of the attacks like, e.g., timings of the clock glitches. We also identify suitable target instructions in the assembly of the target implementation to realize our concrete attacks. Finally, we give information on the results of our experiments. Our positive results show that instruction skips are a realistic threat.

Chapter 9 In this chapter, we conclude and address open questions for future work.

Chapter 2.

Mathematical background

In this chapter, we introduce our notation and the required mathematical background. As part of this, we collect material on PBC that is scattered over different sources like, e.g., [Sil09; BSS05; Gal12; JN09]. In Section 2.1 we introduce our notation and start with some basic definitions mainly based on [Sil09]. In Section 2.2 we give background information on elliptic curves with focus on applications in PBC. Finally, in Section 2.3, we define cryptographic pairings.

2.1. Basic definitions and notation

In this section, we give some basic definitions and fix our notation. With respect to the basic notation we mostly adhere to [Lan02] and only introduce notation when it deviates from [Lan02]. In the glossary on page 151 we give an overview of the notation.

2.1.1. Sets and integers

Let S be a set. With $\#S$ we denote the cardinality of S. If S is ordered we write $[a, b]$ for the closed interval from a to b.

For $x, n \in \mathbb{N}$ with $x < 2^n$ we denote the zero-padded n-bit binary representation of x with $\lceil x \rceil_2^n \in \{0,1\}^n$. For $a, b \in \{0,1\}^*$, we write $a.b$ for the concatenation of a and b. For $n_1, \ldots, n_k \in \mathbb{N}$ we write $\gcd(n_1, \ldots, n_k)$ for the greatest common divisor of $n_1, \ldots, n_k$. For $n, m \in \mathbb{N}$, we write $n|m$ if n divides m, $n \nmid m$ if n does not divide m, and $n \parallel m$ if $n|m$ but $n^2 \nmid m$.

Definition 2.1. *Let $q \in \mathbb{N}$, $a \in \mathbb{Z}$, and $\alpha_0, \ldots, \alpha_{n-1} \in [-\lfloor (q-1)/2 \rfloor, \lceil (q-1)/2 \rceil]$ such that $a = \sum_{i=0}^{n-1} \alpha_i q^i$. Then we define the **weight** of a in base q as $\mathrm{w}_q(a) = \sum_{i=0}^{n-1} |\alpha_i|$.*

We call an element $a \in \mathbb{Z}$ light or heavy with respect to q if $\mathrm{w}_q(a)$ is small or large, respectively. The *Hamming weight* of a is defined as $\mathrm{HW}(a) = \mathrm{w}_2(a)$.

2.1.2. Rings and fields

We use blackboard bold letters $\mathbb{K}$ and $\mathbb{L}$ for arbitrary fields and $\mathbb{F}_q$ for the field with q elements. With $\overline{\mathbb{K}}$ we denote the algebraic closure of a field $\mathbb{K}$, and with $\mathbb{K}^+$

and $\mathbb{K}^*$ we denote the additive and the multiplicative subgroup of $\mathbb{K}$. For a rational function $f = g/h \in \mathbb{K}(x_1, \ldots, x_n)$ with polynomials $g, h \in \mathbb{K}[x_1, \ldots, x_n]$ we define the degree of f as $\deg(f) = \max\{\deg(g), \deg(h)\}$, and with $\mathrm{mon}(f)$ we denote the monomials that occur in f. We write $\mu_n \subseteq \overline{\mathbb{K}}$ for the n-th *roots of unity* in $\overline{\mathbb{K}}$, i.e., the roots of $x^n - 1$. With $\Phi_n(x)$ we denote the n-th cyclotomic polynomial.

With σ_q we denote the *Frobenius automorphism* over $\mathbb{F}_q$. For $f \in \overline{\mathbb{F}}_q[x_1, \ldots, x_n]$ we write $f^q(x_1, \ldots, x_n)$ for the reduction of f modulo σ_q, i.e., for the polynomial where σ_q is applied to each coefficient of $f(x_1, \ldots, x_n)$.

The extension $\mathbb{F}_{q^k}/\mathbb{F}_q$ is a $\mathbb{F}_q$ vector space of dimension k and we assume that, for a given basis, elements in $\mathbb{F}_{q^k}/\mathbb{F}_q$ are represented as coefficient vectors over $\mathbb{F}_q$. Let $q = p^k$ for a prime p and $a \in \mathbb{F}_q$ with representation $(a_0, \ldots, a_{k-1}) \in \mathbb{F}_p^k$. Let b be the LSB of a_0. Then we define $(-1)^b$ as the sign of a. For $p = 2$, we write $\sqrt{a} \in \mathbb{F}_q$ for the unique square root of a. For $p > 2$, we write $\sqrt{a} \in \mathbb{F}_{q^2}$ for the positive square root of a. For any $i \in \mathbb{N}$ we also write $a^{p^{-i}}$ to denote the unique element b in $\mathbb{F}_q$ with $b^{p^i} = a$.

2.1.3. Algebraic sets

We write $\mathbb{A}^n(\mathbb{K}) = \mathbb{K}^n$ for the n-dimensional affine space over $\mathbb{K}$, and with $\mathbb{P}^n(\mathbb{K})$ we denote the associated projective space. We abbreviate $\mathbb{A}^n(\overline{\mathbb{K}})$ with $\mathbb{A}^n$ and $\mathbb{P}^n(\overline{\mathbb{K}})$ with $\mathbb{P}^n$. Usually, we use capital letters for elements in $\mathbb{A}^n$ and $\mathbb{P}^n$ and call them points. For a finite field $\mathbb{F}_q$ we define the q-th power *Frobenius map* as

$$\begin{array}{rcl} \pi_q : \mathbb{A}^n(\overline{\mathbb{F}}_q) & \to & \mathbb{A}^n(\overline{\mathbb{F}}_q) \\ (a_1, \ldots, a_n) & \mapsto & (a_1^q, \ldots, a_n^q). \end{array} \tag{2.1}$$

For $P \in \mathbb{A}^2$ we write $x(P)$ and $y(P)$ for the x-coordinate and the y-coordinate of P, respectively. Furthermore, with $(X : Y : Z) \in \mathbb{P}^2$ we denote the equivalence class that is associated with $(X/Z, Y/Z) \in \mathbb{A}^2$.

Let $\mathcal{F} \subseteq \mathbb{K}[x_1, \ldots, x_n]$. The *affine algebraic set* of $\mathcal{F}$ is defined as

$$V(\mathcal{F}) = \{P \in \mathbb{A}^n \mid f(P) = 0 \text{ for all } f \in \mathcal{F}\}.$$

Similarly, we define the projective algebraic set $V(\mathcal{F}) \subseteq \mathbb{P}^n$ for a set of homogeneous polynomials $\mathcal{F}$. We also abbreviate $V(\{f_1, \ldots, f_m\})$ with $V(f_1, \ldots, f_m)$. Let $\mathbb{L} \subseteq \overline{\mathbb{K}}$ and $\mathcal{F} \subseteq \mathbb{K}[x_1, \ldots, x_n]$ with $X = V(\mathcal{F})$. We define the $\mathbb{L}$-*rational points* of X as $X(\mathbb{L}) = X \cap \mathbb{A}^n(\mathbb{L})$. For an irreducible algebraic set X defined over $\mathbb{K}$, we write $\mathbb{K}(X)$ for the *function field* of X.

Now we introduce the Weil restriction of scalars, a formalization of writing a polynomial equation over $\mathbb{F}_{q^k}$ in n variables as a system of polynomial equations over $\mathbb{F}_q$ in kn variables. We use Weil restriction of scalars in our analysis of FAs on pairings in Chapter 5 and Chapter 6. The following definition is based on Definition 5.7.3 of [Gal12]:

Definition 2.2. *Let $\{\theta_1, \ldots, \theta_k\}$ be a basis for $\mathbb{F}_{q^k}$ as a $\mathbb{F}_q$ vector space. Let $x_1, \ldots, x_n$ be coordinates for $\mathbb{A}^n$ and $x_{1,1}, \ldots, x_{1,k}, \ldots, x_{n,1}, \ldots, x_{n,k}$ be coordinates for $\mathbb{A}^{kn}$. For $i \in [1, n]$ define $\phi_i : \mathbb{A}^k \to \mathbb{A}^1$ by $\phi_i(x_{i,1}, \ldots, x_{i,k}) = \sum_{j=0}^{k} x_{i,j}\theta_j$ and*

$$\phi : \mathbb{A}^{kn} \to \mathbb{A}^n$$
$$(x_{1,1}, \ldots, x_{1,k}, \ldots, x_{n,1}, \ldots, x_{n,k}) \mapsto (\phi_1(x_{1,1}, \ldots, x_{1,k}), \ldots, \phi_n(x_{n,1}, \ldots, x_{n,k})).$$

Let $S \subseteq \mathbb{F}_{q^k}[x_1, \ldots, x_n]$ and let $X = V(S) \subseteq \mathbb{A}^n$ be an affine algebraic set over $\mathbb{F}_{q^k}$. For each polynomial $f(x_1, \ldots, x_n) \in S$ write $\phi^*(f) = f \circ \phi$ as

$$f_1\theta_1 + f_2\theta_2 + \cdots + f_k\theta_k$$

*with $f_1, \ldots, f_k \in \mathbb{F}_q[x_{1,1}, \ldots, x_{n,k}]$. Define $S' \subseteq \mathbb{F}_q[x_{1,1}, \ldots, x_{n,k}]$ to be the set of all such polynomials over all $f \in S$. The **Weil restriction of scalars** of X with respect to $\mathbb{F}_{q^k}/\mathbb{F}_q$ is the affine algebraic set $Y \subseteq \mathbb{A}^{kn}$ defined by $Y = V(S')$.*

The Weil restriction of scalars preserves $\mathbb{F}_{q^k}$ rational solutions of S from Definition 2.2. The following theorem is a corollary of Theorem 5.7.7 from [Gal12].

Theorem 2.3. *Let $X \subseteq \mathbb{A}^n$ be an affine algebraic set over $\mathbb{F}_{q^k}$. Let $Y \subseteq \mathbb{A}^{kn}$ be the Weil restriction of X. Then ϕ from Definition 2.2 is a bijection between $Y(\mathbb{F}_q)$ and $X(\mathbb{F}_{q^k})$.*

2.2. Elliptic curves

In this section, we give some background information on elliptic curves with focus on applications in cryptography and, especially, pairings. In Section 2.2.1 we define elliptic curves. In Section 2.2.2 we define the group law for elliptic curves. In Section 2.2.3 we give background information on the structure of singular curves that we exploit in our attacks from Chapter 7. In Section 2.2.4, we define twists because they have important applications in PBC. In Section 2.2.5 we provide the necessary background information on the structure of torsion subgroups of elliptic curves that is useful to understand implementations of PBC.

2.2.1. Weierstrass equations

To simplify matters, we restrict to fields $\mathbb{K}$ of characteristic not equal to 2 or 3 from now on. For the general case, we refer to [Sil09]. For $a_4, a_6 \in \mathbb{K}$ a homogeneous equation of the form

$$Y^2Z = X^3 + a_4XZ^2 + a_6Z^3 \tag{2.2}$$

is called *short Weierstrass equation* defined over $\mathbb{K}$.

Based on Section III.1 of [Sil09] we define:

Definition 2.4. *For a short Weierstrass equation of the form* (2.2) *define the* ***discriminant*** $\Delta = -16(4a_4^3 + 27a_6^2)$. *A* ***singular*** *Weierstrass equation is a Weierstrass equation with* $\Delta = 0$. *A Weierstrass equation with* $\Delta \neq 0$ *is called* ***smooth****. The* ***j-invariant*** *of a smooth Weierstrass equation is defined as* $j = -1728(4a_4)^3/\Delta$.

For smooth Weierstrass equations, we define:

Definition 2.5. *An* ***elliptic curve*** *is the projective algebraic set* $E \subseteq \mathbb{P}^2$ *of a smooth Weierstrass equation of the form* (2.2). *We write* $E/\mathbb{K}$ *to denote that the Weierstrass equation of* E *is defined over* $\mathbb{K}$.

Let E be an elliptic curve defined by a Weierstrass equation (2.2). With $\mathcal{O} := (0 : 1 : 0)$ we denote *the point at infinity* of E. If we de-homogenize (2.2) with respect to Z we obtain

$$y^2 = x^3 + a_4x + a_6. \tag{2.3}$$

Hence, we can consider E as the affine algebraic set defined by (2.3) together with the additional point $\mathcal{O}$. To ease notation we use the affine representation (2.3) of E whenever possible.

For finite fields, we can bound the number of $\mathbb{F}_q$-rational points of a curve (cf. Chapter V, Theorem 1.1 of [Sil09]):

Theorem 2.6 (Hasse)**.** *Let* $E/\mathbb{F}_q$ *be an elliptic curve. Then*

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

Now, we define a quantity that is often used to characterize elliptic curves over finite fields (cf. Chapter V, Remark 2.6 of [Sil09]):

Definition 2.7. *Let* $E/\mathbb{F}_q$ *be an elliptic curve with*

$$\#E(\mathbb{F}_q) = q + 1 - t$$

for $t \in \mathbb{Z}$. *We call* t *the* ***trace of Frobenius*** *of* $E(\mathbb{F}_q)$.

Based on the trace of Frobenius we distinguish two types of elliptic curves according to the following definition (cf. Definition 9.11.3 of [Gal12]):

Definition 2.8. *Let* $E/\mathbb{F}_q$ *be an elliptic curve with* $\#E(\mathbb{F}_q) = q + 1 - t$. *Then* E *is called* ***ordinary*** *if* $\gcd(q, t) = 1$ *and* ***supersingular*** *if* $\gcd(q, t) > 1$.

Because the DLOG problem is sub-exponential on supersingular curves [MOV93], they are avoided in standard ECC. But in PBC they are frequently used (see also Definition 3.1, and Theorem 3.4 in Chapter 3).

2.2.2. Group law

The following operation turns an elliptic curve E into an abelian group with neutral element $\mathcal{O}$ (cf. Section III.2 of [Sil09]):

Definition 2.9. *Let E be an elliptic curve given by a Weierstrass equation* (2.3)*. Let $P_1, P_2 \in E$ with $\mathcal{O} \notin \{P_1, P_2\}$, $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$.*

1. *Define $-P_1 = (x_1, -y_1)$.*
2. *Define $P_1 + \mathcal{O} = \mathcal{O} + P_1 = P_1$.*
3. *If $P_1 = -P_2$ define $P_1 + P_2 = \mathcal{O}$.*
4. *If $P_1 \neq -P_2$*
   a) *If $P_1 = P_2$ set*
   $$\lambda_{P_1,P_1} = \frac{3x_1^2 + a_4}{2y_1}. \tag{2.4}$$
   b) *If $P_1 \neq P_2$ set*
   $$\lambda_{P_1,P_2} = \frac{y_2 - y_1}{x_2 - x_1}.$$

   Set $x_3 = \lambda_{P_1,P_2}^2 - x_1 - x_2$, and $y_3 = \lambda_{P_1,P_2}(x_1 - x_3) - y_1$. Then define $P_1 + P_2 = (x_3, y_3)$.

Let $n \in \mathbb{Z}$. With $[n] : E \to E$, we denote the multiplication-by-n map that maps a point $P \in E$ to $nP = P + \cdots + P$, n times. We call the computation of nP also elliptic curve scalar multiplication (ECSM).

Remark 2.10. An implementation of ECSM that implements the group operation like in Definition 2.9 does not use the parameter a_6 of (2.3). We exploit this fact in our attack of Chapter 7.

2.2.3. Singular curves

Singular cubic curves, i.e., curves with a singular Weierstrass equation (cf. Definition 2.4) play the central role of our attack in Chapter 7. A singular curve contains singular points where the partial derivatives of (2.3) vanish simultaneously. For a curve E given by a Weierstrass equation, we denote the set of non-singular points with E_{ns}. Note that by definition, $E_{ns} = E$ for elliptic curves.

In Definition 2.9 we introduced a group law for the points on an elliptic curve. The non-singular points E_{ns} of a singular curve define also an abelian group under this operation. The following theorem describes the structure of this group. It summarizes Theorem 2.29 and Theorem 2.30 of [Was03] and adapts Theorem 2.30 of [Was03] to curves in short Weierstrass form (see also Section III.2, Proposition 2.5 of [Sil09]).

Theorem 2.11. *Let $E/\mathbb{F}_q$ be a singular curve defined by a short Weierstrass equation* (2.3).

1. *If $a_4 = a_6 = 0$, then E has a singular point at $S = (0,0)$ and the map $\phi^+ : E_{ns}(\mathbb{F}_q) \to \mathbb{F}_q^+$ defined by*

$$\mathcal{O} \mapsto 0 \quad \text{and} \quad (x,y) \mapsto \frac{x}{y}$$

   is an isomorphism of groups.

2. *If $a_4 \neq 0$ define $x_S = -3a_6/(2a_4)$. Then E has a singular point at $S = (x_S, 0)$. Furthermore, with $\alpha = \sqrt{3x_S}$ define the map $\phi^* : E_{ns}(\mathbb{F}_q) \to \mathbb{F}_q^*(\alpha)$ by*

$$\mathcal{O} \mapsto 1 \quad \text{and} \quad (x,y) \mapsto \frac{y - \alpha(x - x_S)}{y + \alpha(x - x_S)}.$$

   a) If $\alpha \in \mathbb{F}_q$, then ϕ^* is an isomorphism of the groups $E_{ns}(\mathbb{F}_q)$ and $\mathbb{F}_q^*$.

   b) If $\alpha \notin \mathbb{F}_q$, then ϕ^* is an isomorphism of $E_{ns}(\mathbb{F}_q)$ and $\{u + \alpha v \mid u, v \in \mathbb{F}_q, u^2 - 3x_S v^2 = 1\}$ viewed as a multiplicative group.

Proof. For case 1 note that the partial derivatives $3x^2$ and $2y$ of $x^3 - y^2$ vanish simultaneously at $S = (0,0)$ and hence S is singular by definition. According to Theorem 2.29 of [Was03] the map ϕ^+ is an isomorphism of groups.

For case 2a and case 2b we see that the partial derivatives $2y$ and $3x^2 + a_4$ of (2.3) vanish at $S = (x_S, 0)$ and hence S is singular. Now we translate (2.3) such that the singular point is at $(0,0)$. By expanding $y^2 = (x + x_S)^3 + a_4(x + x_S) + a_6$ and applying the identity $4a_4^3 + 27a_6^2 = 0$ from the discriminant of singular curves we obtain $y^2 = x^3 + 3x_S x^2$. Then, it follows from Theorem 2.30 of [Was03] that ϕ^* is an isomorphism of groups. □

2.2.4. Twists

Twists play an important role for the efficient implementation of PBC and hence, they also need to be considered for implementation attacks. Furthermore, twists also have destructive applications in FAs on ECSM. Therefore, we give some background here. For more details about twists, we refer to [HSV06].

The following theorem shows when two elliptic curves are isomorphic in a proper extension field:

Theorem 2.12 (Chapter III, Proposition 1.4 (b) of [Sil09])**.** *Let $E/\mathbb{K}$ and $E'/\mathbb{K}$ be two elliptic curves given by a Weierstrass equation* (2.3)*. Then E and E' are isomorphic over $\overline{\mathbb{K}}$ if and only if they have the same j-invariant (cf. Definition 2.4).*

For elliptic curves over finite fields $\mathbb{F}_q$, we give a more precise definition that includes the extension $\mathbb{F}_{q^d}/\mathbb{F}_q$ where two curves with the same j-invariant become isomorphic:

Definition 2.13 (Definition II.7 of [Ver09]). *Let $E/\mathbb{F}_q$ and $E'/\mathbb{F}_q$ be two elliptic curves, then E' is called a **degree** d **twist** of E, if there exists an isomorphism $\psi : E' \to E$ defined over $\mathbb{F}_{q^d}$ and d is minimal.*

There is only a limited number of twists for a given curve:

Theorem 2.14 (Proposition 6 of [HSV06]). *Let $E/\mathbb{F}_q$ be an elliptic curve with j-invariant j and let $p \geq 5$ be the characteristic of $\mathbb{F}_q$. Then the set of twists of E is canonically isomorphic with $\mathbb{F}_q^*/(\mathbb{F}_q^*)^d$ where $d = 2$ if $j \notin \{0, 1728\}$, $d = 4$ if $j = 1728$, and $d = 6$ if $j = 0$.*

Note that the possible degrees of the twists correspond to the orders of elements in $\mathbb{F}_q^*/(\mathbb{F}_q^*)^d$ and hence they divide d. We see that for fields of large characteristic, only special curves with $j \in \{0, 1728\}$ have twists of degree $d > 2$.

For an explicit list of the twist of a curve, including the corresponding isomorphism, we refer to Section 1.4 of [Ver09] or to Section IV.A of [HSV06].

2.2.5. Torsion points

Let E be an elliptic curve defined over a field $\mathbb{K}$. The *n-torsion points* of E are defined as

$$E[n] = \ker([n]) = \{P \in E \mid nP = \mathcal{O}\}.$$

Furthermore, define $E(\mathbb{K})[n] = E[n] \cap E(\mathbb{K})$. Torsion subgroups of an elliptic curve are important in the context of PBC because they define the domain of pairings. Therefore, we give some background information on torsion points and their efficient representation. We will see that twists play an important role in this context. From now on, to emphasize that an integer is prime we denote it with r, while we use n for general integers.

The following theorem describes the structure of $E[n]$ in the case of curves over finite fields (cf. Chapter III, Corollary 6.4 (b) of [Sil09]):

Theorem 2.15. *Let $E/\mathbb{F}_q$ be an elliptic curve an let $n \in \mathbb{Z}$ with $\gcd(q, n) = 1$. Then*

$$E[n] \simeq \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}.$$

Hence, for a prime r, $E[r]$ is a 2-dimensional $\mathbb{Z}/r\mathbb{Z}$ vector space.

We now show which extension $\mathbb{F}_{q^k}/\mathbb{F}_q$ we need to consider such that $E[n]$ is contained in $E(\mathbb{F}_{q^k})$. Therefore, we define:

Definition 2.16. *For $\mathbb{F}_q$ and $n \in \mathbb{N}$ define the **embedding degree** of $\mathbb{F}_q$ (respectively q) and n to be the smallest positive integer $k = k(q, n) \in \mathbb{N}$ such that $n | q^k - 1$.*

Hence, the embedding degree $k(q, n)$ defines the smallest extension field $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$ such that the n-th roots of unity μ_n are contained in $\mathbb{F}_{q^k}$. The following theorem shows that for primes r, the embedding degree also defines the extension field that we have to consider to obtain $E[r]$:

Theorem 2.17 (Theorem 1 of [BK98]). *Let $E/\mathbb{F}_q$ be an elliptic curve and let r be a prime with $r|\#E(\mathbb{F}_q)$ and $r \nmid q-1$. Then $E[r] \subseteq E(\mathbb{F}_{q^k})$ if and only if $r|q^k-1$.*

On E, the Frobenius map π_q from (2.1) is an endomorphism of groups called the *Frobenius endomorphism*. Theorem 2.15 shows that $E[n]$ is the direct sum of two cyclic subgroups of order n. Based on π_q, we define two subgroups that provide efficient representations of $E[n]$:

Definition 2.18. *Let $E/\mathbb{F}_q$ be an elliptic curve and $n \in \mathbb{N}$ with $\gcd(n,q)=1$. We define:*

$$\mathbb{G}_1(E,n,q) = E[n] \cap \ker(\pi_q - [1]) = E(\mathbb{F}_q)[n]$$
$$\mathbb{G}_2(E,n,q) = E[n] \cap \ker(\pi_q - [q]).$$

Usually, we write $\mathbb{G}_1 = \mathbb{G}_1(E,n,q)$ and $\mathbb{G}_2 = \mathbb{G}_2(E,n,q)$ if the definitions of E, n, and q are clear from the context. From Theorem 2.15 we see that we can represent $E[n]$ with $\mathbb{G}_1$ and $\mathbb{G}_2$ as long as $\mathbb{G}_1 \neq \mathbb{G}_2$. A sufficient condition is given as follows:

Theorem 2.19. *Let $E/\mathbb{F}_q$ be an elliptic curve, r a prime with $\gcd(r,q)=1$ and $r|\#E(\mathbb{F}_q)$. Furthermore, let $k>1$ be the embedding degree of q and r. For $\mathbb{G}_1$ and $\mathbb{G}_2$ from Definition 2.18 it holds that*

$$E[r] = \mathbb{G}_1 \oplus \mathbb{G}_2.$$

Proof. See Lemma 1.60 of [Nae09]. □

Because π_q is an endomorphism of groups and because $E[r] \simeq (\mathbb{Z}/r\mathbb{Z})^2$ we denote $\mathbb{G}_1$ and $\mathbb{G}_2$ also as the 1-eigenspace and the q-eigenspace of the Frobenius endomorphism. Now we show that for ordinary curves (cf. Definition 2.8), $\mathbb{G}_2$ can efficiently be represented based on twists. We first need the following theorem:

Theorem 2.20 (Theorem II.10 of [Ver09]). *Let $E/\mathbb{F}_q$ be an ordinary elliptic curve admitting a twist of degree d. Assume that a prime $r>6$ satisfies $r \parallel \#E(\mathbb{F}_q)$ and $r^2 \parallel \#E(\mathbb{F}_{q^d})$. Then there exists a unique twist E' of degree d such that $r \parallel \#E'(\mathbb{F}_q)$.*

This provides a different representation of $\mathbb{G}_2$ (cf. Section V of [HSV06]):

Theorem 2.21. *Let $E/\mathbb{F}_q$ and r be as in Theorem 2.20 and $\mathbb{G}_2 \subseteq E[r]$ be as in Definition 2.18. Let E' be the unique twist from Theorem 2.20 with isomorphism $\psi : E' \to E$. Then $\mathbb{G}_2 = \psi(E'(\mathbb{F}_q)[r])$.*

Remark 2.22. Now let $E/\mathbb{F}_q$ be an elliptic curve with a twist E' of degree d. Furthermore let $r>6$ be a prime with $r \parallel \#E(\mathbb{F}_q)$ and let k be the embedding degree of q and r such that $d|k$. Based on Theorem 2.20 and Theorem 2.21 we obtain the representation $\mathbb{G}_2 = \psi(E'(\mathbb{F}_{q^{k/d}})[r])$. We see that with a twist of degree d, we can reduce the size of the representation of $\mathbb{G}_2 \subseteq E(\mathbb{F}_{q^k})$ by a factor of d.

2.3. Pairings

In cryptography, a pairing is a non-degenerate, bilinear map $\mathrm{e} : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$ with groups $\mathcal{G}_1, \mathcal{G}_2$, and $\mathcal{G}_T$. We are only interested in pairings that are defined based on elliptic curves $E/\mathbb{K}$ with $\mathcal{G}_1, \mathcal{G}_2 \subset E$ and $\mathcal{G}_T \subset \mathbb{K}^*$. To define pairings, we introduce the concept of divisors in Section 2.3.1. Then, in Section 2.3.2, we define the Tate pairing with the help of divisors. Because pairings are interesting for application in cryptography, a lot of research was done to find pairings that can be computed more efficiently. Examples are the Ate pairing [HSV06], R-Ate pairing [LLP09], Eta pairing [Bar+07], and optimal pairings [Ver10]. We present two examples, the reduced Tate pairing and the reduced Ate pairing in Section 2.3.3.

2.3.1. Divisors on elliptic curves

In this section, we introduce a useful tool to describe functions on elliptic curves.

Definition 2.23. *Let E be an elliptic curve. A **divisor** on E is a finite formal sum*

$$D = \sum_{P \in E} n_P \, [P] \tag{2.5}$$

with $n_P \in \mathbb{Z}$. Furthermore

1. *The **support** of D is defined as*

$$\mathrm{supp}(D) = \{P \in E \mid n_P \neq 0\}.$$

2. *The **degree** of D is defined as*

$$\deg(D) = \sum_{P \in E} n_P.$$

3. *The **sum** of D is defined as*

$$\mathrm{sum}(D) = \sum_{P \in E} n_P P.$$

To define the divisor of a function on an elliptic curve, we need the definition of the *order* $\mathrm{ord}_P(f) \in \mathbb{Z}$ of a function $f \in \overline{\mathbb{K}}(E)$ at a point $P \in E$. The exact definition requires additional background and therefore, we refer to Section II.1 of [Sil09]. We say that f has a *zero* of order n at P if $n = \mathrm{ord}_P(f) > 0$ and we say that f has a *pole* of order n at P if $n = \mathrm{ord}_P(f) < 0$.

Definition 2.24. *For an elliptic curve $E/\mathbb{K}$ and a function $f \in \overline{\mathbb{K}}(E)^*$ define the divisor of f as*

$$\operatorname{div}(f) = \sum_{P \in E} \operatorname{ord}_P(f)(P).$$

A divisor D on $E/\mathbb{K}$ is called a *principal divisor* if it is the divisor of a function $f \in \overline{\mathbb{K}}(E)^*$. Two divisors D_1 and D_2 on E are *equivalent*, written as $D_1 \sim D_2$, if $D_1 - D_2$ is a principal divisor. We now give a distinguishing property of principal divisors:

Theorem 2.25 (Corollary 3.5 of [Sil09])**.** *Let $E/\mathbb{K}$ be an elliptic curve an D be a divisor on E. Then D is a principal divisor if and only if* $\deg(D) = 0$ *and* $\operatorname{sum}(D) = \mathcal{O}$.

The following theorem shows that principal divisors define functions up to constant multiples:

Theorem 2.26 (Corollary 7.7.13 of [Gal12])**.** *Let $E/\mathbb{K}$ be an elliptic curve and $f, g \in \mathbb{K}(E)^*$. Then* $\operatorname{div}(f) = \operatorname{div}(g)$ *if and only if $f = cg$ for some $c \in \mathbb{K}^*$.*

Together with the following theorem this shows that divisors are a useful tool to describe functions on elliptic curves:

Theorem 2.27 (Lemma 7.7.4 of [Gal12])**.** *Let $E/\mathbb{K}$ be an elliptic curve and let $f, g \in \mathbb{K}(E)^*$. Then*

1. $\operatorname{div}(fg) = \operatorname{div}(f) + \operatorname{div}(g)$
2. $\operatorname{div}(1/f) = -\operatorname{div}(f)$
3. $\operatorname{div}(f^n) = n \operatorname{div}(f)$ *for $n \in \mathbb{Z}$.*

Now we define what it means to evaluate a function at a divisor:

Definition 2.28. *Let $E/\mathbb{K}$ be an elliptic curve and D a divisor on E as in* (2.5) *of Definition 2.23. Furthermore, let $f \in \overline{\mathbb{K}}(E)^*$ with* $\operatorname{supp}(\operatorname{div}(f)) \cap \operatorname{supp}(D) = \emptyset$. *Then define*

$$f(D) = \prod_{P \in E} f(P)^{n_P}.$$

2.3.2. The Tate pairing

Now we are able to define the Tate pairing with the help of divisors:

Definition 2.29 (Definition II.15 of [Ver09])**.** *Let $E/\mathbb{F}_q$ be an elliptic curve and $n \in \mathbb{N}$ with $n | \#E(\mathbb{F}_q)$ and* $\gcd(n, q) = 1$. *Let k be the embedding degree of q and n. Then the* ***Tate pairing*** *is the map*

$$\begin{aligned} \mathrm{t}_n : E(\mathbb{F}_q)[n] \times E(\mathbb{F}_q)/nE(\mathbb{F}_{q^k}) &\to \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^n \\ (P, Q) &\mapsto f_{n,P}(D_Q), \end{aligned}$$

where $f_{n,P} \in \overline{\mathbb{F}}_q(E)$ *with* $\operatorname{div}(f_{n,P}) = n\,[P] - n\,[\mathcal{O}]$ *and* $D_Q \sim [Q] - [\mathcal{O}]$ *such that* $\operatorname{supp}(D_Q) \cap \{\mathcal{O}, P\} = \emptyset$.

The Tate pairing is well-defined (cf. Lemma 26.3.2 of [Gal12]), and the following properties make the Tate pairing interesting for cryptographic applications:

Theorem 2.30 (Theorem II.16 of [Ver09])**.** *The Tate pairing from Definition 2.29 satisfies the following properties:*

1. Bilinearity: *For* $P_1, P_2 \in E(\mathbb{F}_{q^k})[n]$ *and* $Q_1, Q_2 \in E(\mathbb{F}_{q^k})/nE(\mathbb{F}_{q^k})$ *it holds that*
$$\mathrm{t}_n(P_1 + P_2, Q_1) = \mathrm{t}_n(P_1, Q_1)\,\mathrm{t}_n(P_2, Q_1) \text{ and}$$
$$\mathrm{t}_n(P_1, Q_1 + Q_2) = \mathrm{t}_n(P_1, Q_1)\,\mathrm{t}_n(P_1, Q_2).$$

2. Non-degeneracy:
   a) *For all* $P \in E(\mathbb{F}_{q^k})[n]$ *with* $P \neq \mathcal{O}$ *there exists a* $Q \in E(\mathbb{F}_{q^k})$ *such that* $\mathrm{t}_n(P, Q) \notin (\mathbb{F}_q^*)^n$.
   b) *For all* $Q \in E(\mathbb{F}_{q^k})$ *with* $Q \notin nE(\mathbb{F}_{q^k})$ *there exists a* $P \in E(\mathbb{F}_{q^k})[n]$ *such that* $\mathrm{t}_n(P, Q) \notin (\mathbb{F}_q^*)^n$.

The Tate pairing is defined based on a function with divisor satisfying $n\,[P] - n\,[\mathcal{O}]$. Now we define a family of functions that we use in Section 3.2.2 to iteratively construct a function with this divisor:

Definition 2.31. *Let* $E/\mathbb{K}$ *and* $P \in E$*. With* $n \in \mathbb{Z}$ *we define the* ***Miller function*** $\mathrm{mil}_{n,P} \in \overline{\mathbb{K}}(E)$ *for* n *and* P *to be the unique normalized function with divisor*

$$\operatorname{div}(\mathrm{mil}_{n,P}) = n\,[P] - [nP] - (n-1)\,[\mathcal{O}]\,.$$

The existence of such a function follows from Theorem 2.25. For the definition of a normalized function we refer to Definition 4 of [Mil04].

Lemma 2.32. *Let* E *be an elliptic curve and* $P \in E[n]$*. Then* $\operatorname{div}(\mathrm{mil}_{n,P}) = n\,[P] - n\,[\mathcal{O}]$.

Proof. Because $P \in E[n]$ it holds that $nP = \mathcal{O}$. Then, the statement follows directly from the definition of $\mathrm{mil}_{n,P}$. □

2.3.3. Reduced pairings

We see in Definition 2.29 that the Tate pairing maps to cosets of $(\mathbb{F}_{q^k}^*)^n$ and not to unique representations in $\mathbb{F}_{q^k}^*$. For the *reduced Tate pairing*, the result of the Tate pairing is exponentiated with $(q^k - 1)/n$. This maps elements in $(\mathbb{F}_{q^k}^*)^n$ to 1 and we obtain unique representations in the n-th roots of unity $\mu_n \subseteq \mathbb{F}_{q^k}^*$.

Reduced Tate pairing

We define a version of the reduced Tate pairing that is typically used in cryptographic implementations:

Definition 2.33. *Let $E/\mathbb{F}_q$ be an elliptic curve and $r > 4$ be a prime with $r|\#E(\mathbb{F}_q)$ and $\gcd(r, q) = 1$. Furthermore, let $k > 1$ be the embedding degree of q and r. Then we define the reduced Tate pairing as*

$$\hat{\mathrm{t}}_r : E(\mathbb{F}_q)[r] \times E(\mathbb{F}_{q^k})[r] \to \mu_r$$
$$(P, Q) \mapsto \begin{cases} 1 & \text{if } Q \in E(\mathbb{F}_q) \\ \mathrm{mil}_{r,P}(Q)^{(q^k-1)/r} & \text{else.} \end{cases}$$

Note that compared to Definition 2.29, we restrict to primes r and a $\mathbb{F}_q$-rational first argument $P \in E(\mathbb{F}_q)$. Furthermore, we use the normalized instantiation $\mathrm{mil}_{r,P}$ for a function with divisor $r\,[P] - r\,[\mathcal{O}]$. Finally, the domain of the second argument of the reduced pairing is $E(\mathbb{F}_{q^k})[r]$ instead of $E(\mathbb{F}_q)/rE(\mathbb{F}_{q^k})$ and $\mathrm{mil}_{r,P}$ is evaluated at the point Q and not at the divisor D_Q.

Theorem 2.34. *The reduced Tate pairing $\hat{\mathrm{t}}_r$ from Definition 2.33 defines a non-degenerate, bilinear pairing if $r^3 \nmid \#E(\mathbb{F}_{q^k})$.*

We prove the theorem in a sequence of lemmas that are of independent interest. The first lemma shows that we can use $E[r]$ to represent $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$ (cf. Exercise 26.3.10 of [Gal12]):

Lemma 2.35. *Let $E/\mathbb{F}_q$ be an elliptic curve, and r be a prime with $\gcd(r, q) = 1$ and $r|\#E(\mathbb{F}_q)$. Furthermore, let $k > 1$ be the embedding degree of q and r. If $r^3 \nmid \#E(\mathbb{F}_{q^k})$ it holds that $E[r]$ is a set of representatives for $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$.*

Proof. We first show that $E(\mathbb{F}_{q^k})$ contains no points of order r^2. Because $k > 1$ it follows from Theorem 2.17 that $E[r] \subseteq E(\mathbb{F}_{q^k})$. With $\gcd(r, q) = 1$ and $E[r] \subseteq E(\mathbb{F}_{q^k})$ it follows from Theorem 2.15 that $E(\mathbb{F}_{q^k})$ has a subgroup isomorphic to $\mathbb{Z}/r\mathbb{Z} \oplus \mathbb{Z}/r\mathbb{Z}$. If we assume that $E(\mathbb{F}_{q^k})$ has a point of order r^2, then $E(\mathbb{F}_{q^k})$ has a subgroup isomorphic to $\mathbb{Z}/r^2\mathbb{Z} \oplus \mathbb{Z}/r\mathbb{Z}$ of size r^3. This is a contradiction because $r^3 \nmid \#E(\mathbb{F}_{q^k})$.

Now we show that every coset of $rE(\mathbb{F}_{q^k})$ contains exactly one element of $E[r]$. Because $E[r] \subseteq E(\mathbb{F}_{q^k})$ it holds that $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$ and $E[r]$ are of the same size. Hence, it remains to show that the canonical projection from $E[r]$ to $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$ is injective, or equivalently, that it has trivial kernel. Assume there exists a $P \in E[r] \cap rE(\mathbb{F}_{q^k})$ with $P \neq \mathcal{O}$. Then there exists a $Q \in E(\mathbb{F}_{q^k})$ with $rQ = P$. Hence, $\mathrm{ord}(Q) \nmid r$ but $\mathrm{ord}(Q)|r^2$. Because $E(\mathbb{F}_{q^k})$ does not contain points of order r^2 and because r is prime this is a contradiction. □

The following lemma shows that we can set $\hat{\mathrm{t}}(P, Q) = 1$ for the case where both arguments P and Q are elements of $E(\mathbb{F}_q)[r]$:

Lemma 2.36. *Let $E/\mathbb{F}_q$ be an elliptic curve and let r be a prime. Let $k > 1$ be the embedding degree of q and r, and $P, Q \in E(\mathbb{F}_q)[r]$. Then $\mathrm{t}_r(P,Q) \in (\mathbb{F}_{q^k}^*)^r$.*

Proof. This is a corollary of [BSS05, Lemma IX.8]. □

The next lemma shows that we can replace the divisor D_Q in Definition 2.29 by the point Q if we restrict the first argument to $E(\mathbb{F}_q)[r]$:

Lemma 2.37. *Let $E/\mathbb{F}_q$ be an elliptic curve, $r > 4$ be a prime with $\gcd(q, r) = 1$ and embedding degree $k > 1$, $P \in E(\mathbb{F}_q)[r]$, and $Q \in E(\mathbb{F}_{q^k}) \backslash E(\mathbb{F}_q)$. Then*

$$\mathrm{t}_r(P,Q)/\operatorname{mil}_{r,P}(Q) \in (\mathbb{F}_{q^k}^*)^r.$$

Proof. This follows from Lemma 26.3.11 of [Gal12]. □

Hence, in the setting of this lemma, we can compute the reduced Tate pairing with only one application of the Miller function $\operatorname{mil}_{r,P}$. In summary, Theorem 2.34 follows from Theorem 2.30, Lemma 2.35, Lemma 2.36, and Lemma 2.37.

Reduced Ate pairing

We finally define the reduced Ate pairing that is often more efficient than the reduced Tate pairing and therefore used in many practical implementations.

Definition 2.38. *Let $E/\mathbb{F}_q$ be an elliptic curve and r be a prime with $r | \#E(\mathbb{F}_q)$ and $\gcd(q, r) = 1$. Furthermore, let $k > 1$ be the embedding degree of q and r. Define $\lambda = q \mod r$. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be as in Definition 2.18. Then the **(reduced) Ate pairing** is defined as*

$$\begin{aligned} \mathrm{ate}_\lambda : \mathbb{G}_2 \times \mathbb{G}_1 &\to \mu_r \\ (P,Q) &\mapsto \begin{cases} 1 & \text{if } Q = \mathcal{O} \\ \operatorname{mil}_{\lambda,P}(Q)^{(q^k-1)/r} & \text{else.} \end{cases} \end{aligned}$$

Note that compared to the reduced Tate pairing, the Miller function $\operatorname{mil}_{r,P}$ is parameterized with an element $P \in \mathbb{G}_2$ while it is evaluated at an element $Q \in \mathbb{G}_1$.

Let t be the trace of Frobenius of $E(\mathbb{F}_q)$. From Definition 2.7, we see that $t - 1 = q \mod r$. Hence, $\lambda = t - 1$ provides a possible parametrization of the Ate pairing. For other choices, we refer to [Ver10]. From $|t| \leq 2\sqrt{q}$ (see Theorem 2.6 and Definition 2.7) and $r \approx q$ it follows that the bit length of t is approximately half the bit length of r. We will see from Algorithm 3.4 for computing $\operatorname{mil}_{r,P}(Q)$ or $\operatorname{mil}_{\lambda,P}(Q)$ that this makes implementations of the reduced Ate pairing potentially more efficient than implementations of the reduced Tate pairing. The following theorem shows when the Ate pairing has the desired properties:

Theorem 2.39 (Theorem II.22 of [Ver09])**.** *Let r, k, and λ be defined as in Definition 2.38. Define $c = (\lambda^k - 1)/r$. Then ate_λ defines a non-degenerate, bilinear pairing if and only if $\gcd(c, r) = 1$.*

The proof in [Ver09] actually shows that $\mathrm{ate}_\lambda(P,Q) = \hat{\mathrm{t}}(Q,P)^\tau$ for a constant $\tau \in \mathbb{Z}$. Hence, the Ate pairing can be considered as a special case of the reduced Tate pairing with arguments restricted to $\mathbb{G}_1$ and $\mathbb{G}_2$.

Chapter 3.

Pairing-based cryptography

In this chapter, we give background information on implementations of PBC. In Section 3.1 we start with some definitions related to the groups that act as domain an codomain of pairings. Furthermore, we present concrete instantiations of such pairing groups. In Section 3.2 we provide algorithms that are frequently used in ECC and PBC. We later refer to these algorithms in order to explain our attacks. To motivate our attacks, we present examples for pairing-based schemes in Section 3.3.

3.1. Pairing groups

From Section 2.3, we know that a pairing is defined based on three groups $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_T$ where $\mathcal{G}_1$ and $\mathcal{G}_2$ are subgroups of an elliptic curve and $\mathcal{G}_T$ is a subgroup of the multiplicative group of a finite field. We start with some useful definitions related to these pairing groups.

3.1.1. Definitions

The literature on PBC typically distinguishes between four different types of pairing groups. The types are defined in various ways (cf. [GPS08; CCS07; CHM10; CM11]). We give a definition that is based on Section A.10.2 of [IEE13]:

Definition 3.1. *For $E/\mathbb{F}_q$ let r be a prime dividing $\#E(\mathbb{F}_q)$ and let k be the embedding degree of q and r. Furthermore, based on $\mathbb{G}_1(E,r,q)$ and $\mathbb{G}_2(E,r,q)$ from Definition 2.18 we define the following four types of pairing groups $(\mathcal{G}_1,\mathcal{G}_2,\mathcal{G}_T)$:*

1. ***Type 1**: E supersingular (cf. Definition 2.8) and $\mathcal{G}_1 = \mathcal{G}_2 = \mathbb{G}_1$.*
2. ***Type 2**: $\mathcal{G}_1 = \mathbb{G}_1$ and $\mathcal{G}_2 = \langle Q \rangle$ for $Q \in E[n] \backslash (\mathbb{G}_1 \cup \mathbb{G}_2)$.*
3. ***Type 3**: $\mathcal{G}_1 = \mathbb{G}_1$ and $\mathcal{G}_2 = \mathbb{G}_2$ or $\mathcal{G}_1 = \mathbb{G}_2$ and $\mathcal{G}_2 = \mathbb{G}_1$.*
4. ***Type 4**: $\mathcal{G}_1 = \mathbb{G}_1$ and $\mathcal{G}_2 = E[r]$.*

From an implementation perspective, Type 3 groups are most important: For example, the Ate pairing from Definition 2.38 is explicitly defined on Type 3 groups. Furthermore, based on Theorem 2.19 and Theorem 2.21, with Type 3 groups we obtain an efficient representation of the torsion points $E[r]$ that also results in more

efficient pairing computations (see [BLS04a; CM11]). Finally, we can transfer the computation of the Tate and the Ate pairing on Type 2 and Type 4 groups to the computation on Type 3 groups (see [KP06; CM11]).

In practice, an elliptic curve based (sub-)group is instantiated based on so-called domain parameters (cf. [ANS99] and [Ant+03]):

Definition 3.2. *ECC **domain parameters** are a tuple $D = (\mathbb{F}_q, E, R, r, c)$ with*

1. *A finite field $\mathbb{F}_q$ defined by a proper description.*
2. *An elliptic curve E defined by two field elements $a_4, a_6 \in \mathbb{F}_q$ and the corresponding Weierstrass equation (2.3).*
3. *A point $R \in E(\mathbb{F}_q)$ of prime order r defined by two field elements $x_R, y_R \in \mathbb{F}_q$ as $R = (x_R, y_R)$*
4. *The order r of R.*
5. *The cofactor c of r defined as $\#E(\mathbb{F}_q) = cr$.*

Now, we can instantiate a tuple of pairing groups $(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T)$ based on ECC domain parameters $D_1 = (\mathbb{F}_q, E_1, R_1, r, c_1)$ and $D_2 = (\mathbb{F}_q, E_2, R_2, r, c_2)$ with $\mathcal{G}_1 \subseteq E_1(\mathbb{F}_q)[r]$, $\mathcal{G}_2 \subseteq E_2(\mathbb{F}_q)[r]$, and $\mathcal{G}_T \subseteq \mathbb{F}_q^*$. The case $E_1 \neq E_2$ occurs for Type 3 groups when we represent $\mathcal{G}_2 = \mathbb{G}_2$ based on Theorem 2.21 such that E_2 is a twist of E_1.

Because we present an invalid point attack in Chapter 7 we define what it means for a point to be valid with respect to given domain parameters:

Definition 3.3. *A point $Q = (x_Q, y_Q)$ is **valid** with respect to domain parameters $D = (\mathbb{F}_q, E, R, r, c)$ if $Q \in \langle R \rangle$ and if Q is of order r.*

3.1.2. Efficient representation of twists

We know from Theorem 2.21 that we can represent $\mathbb{G}_2 \subseteq E(\mathbb{F}_{q^k})$ efficiently based on twists (cf. Definition 2.13) with an isomorphism $\psi : E' \to E$ that induces the isomorphism $\psi : E'(\mathbb{F}_{q^{k/d}})[r] \to \mathbb{G}_2$. In this section, we show how the twist from Theorem 2.21 is typically instantiated because in Chapter 4, we exploit some properties of this instantiation for an SCA.

For a curve E with a twist E' of degree $d \in \{2, 4, 6\}$, $\mathbb{F}_{q^k}$ and ψ are typically defined such that $\mathbb{F}_{q^k} = \mathbb{F}_q[X]/(X^d - a) = \mathbb{F}_{q^{k/d}}(\alpha)$ for $\alpha^d = a$ and

$$\psi(\mathcal{O}) = \mathcal{O} \quad \text{and} \quad \psi(x, y) = (\alpha^2 x, \alpha^3 y). \tag{3.1}$$

Let $y^2 = x^3 + a_4 x + a_6$ and $y^2 = x^3 + a_4' x + a_6'$ be the defining equations of E and E', respectively. Then it holds that

$$a_4 = \alpha^4 a_4' \qquad a_6 = \alpha^6 a_6'. \tag{3.2}$$

For more background, we refer to Section V of [HSV06].

Let $P' \in E'(\mathbb{F}_{q^{k/d}})$ and $P = \psi(P')$ with $P = (x_P, y_P)$ and $P' = (x'_P, y'_P)$. If we write x_P and y_P as vectors with respect to the basis $\{1, \alpha, \cdots, \alpha^{d-1}\}$ of $\mathbb{F}_{q^k}$ as a $\mathbb{F}_{q^{k/d}}$ vector space and with $\alpha^d = a \in \mathbb{F}_q$ we obtain:

d	x_P	y_P
2	$[ax'_P, 0]$	$[0, ay'_P]$
4	$[0, 0, x'_P, 0]$	$[0, 0, 0, y'_P]$
6	$[0, 0, x'_P, 0, 0, 0]$	$[0, 0, 0, y'_P, 0, 0]$.

(3.3)

On the one hand, this sparse representation of elements in $\mathbb{G}_2$ results in efficient implementations of pairings [Beu+10]. On the other hand, we present an SCA in Chapter 4 that exploits exactly this sparseness.

3.1.3. Concrete instantiations of pairing groups

In this section, we provide exemplary instantiations of pairing groups. Therefore, we define two families of elliptic curves. One family is an example for a family of supersingular curves and the other family is an example for a family of ordinary curves (cf. Definition 2.8).

For the following theorem see Section 1.3.1 of [Ver09]:

Theorem 3.4. *Let q and r be primes with $q > 3$ and $q = -1 \mod r$. Then there exists a supersingular elliptic curve E defined over $\mathbb{F}_q$ with $\#E(\mathbb{F}_q) = q + 1 = cr$. The embedding degree (cf. Definition 2.16) of q and r is $k = 2$, and the curve can be given by the equation*

1. *$E : y^2 = x^3 + a_6$ with $a_6 \in \mathbb{F}_q^*$ if $q = 2 \mod 3$,*
2. *$E : y^2 = x^3 + a_4 x$ with $a_4 \in \mathbb{F}_q^*$ if $q = 3 \mod 4$.*

This family of supersingular curves is the standard choice to instantiate Type 1 pairings (cf. Definition 3.1). The following theorem shows that this family of curves attains the largest possible embedding degree for supersingular curves over large prime characteristic fields:

Theorem 3.5. *Let $E/\mathbb{F}_q$ be a supersingular elliptic curve and $q > 3$. Furthermore, let $n | \#E(\mathbb{F}_q)$. Then the embedding degree k of q and n fulfills $k \leq 2$.*

Proof. See Theorem IX.20 of [BSS05]. □

To balance the conjectured exponential complexity of the DLOG problem in $E(\mathbb{F}_q)$ and the sub-exponential complexity of the DLOG problem in $\mathbb{F}_{q^k}^*$, larger embedding degrees are required [FST10]. For the 128 bit security level, $k = 12$ is recommended [FST10]. This motivates the following family of ordinary curves:

Theorem 3.6 (Theorem 2.2 of [Nae09]). *Let $u \in \mathbb{Z}$ be an integer such that*

$$q = q(u) = 36u^4 + 36u^3 + 24u^2 + 6u + 1 \text{ and}$$
$$r = r(u) = 36u^4 + 36u^3 + 18u^2 + 6u + 1$$

are prime numbers. Then there exists an ordinary elliptic curve E defined over $\mathbb{F}_q$ with $\#E(\mathbb{F}_q) = r$. The embedding degree of q and r is $k = 12$, and the curve can be given by the equation

$$E : y^2 = x^3 + a_6 \text{ with } a_6 \in \mathbb{F}_q.$$

The trace of Frobenius (cf. Definition 2.7) over $\mathbb{F}_q$ is given by $t = t(u) = 6u^2 + 1$.

The curves of Theorem 3.6 are also called BN curves because they were first described by Barreto and Naehrig in [BN06].

Corollary 3.7. *Let E be a BN curve from Theorem 3.6. Then E has j-invariant $j = 0$, and admits a twist of degree $d = 6$.*

Proof. That $j = 0$ follows from Definition 2.4 together with the Weierstrass equation of E. Then, with Theorem 2.14 and $q = 1 \mod 6$, it follows that E admits a twist of degree $d = 6$. □

Because BN curves admit a twist of degree $d = 6$ we can represent $\mathbb{G}_2$ already over $\mathbb{F}_{q^2}$ (cf. Remark 2.22). That makes implementations of PBC that are based on BN curves more efficient. Therefore, on the one hand, BN curves are very popular in PBC. On the other hand, in Section 4.3, Section 6.3, and Section 7.2 we introduce attacks that exploit properties of curves with j-invariant 0 like BN curves.

3.2. Algorithms and implementations

In this section, we define algorithms that are used as building blocks for the implementation of PBC. To simplify notation, we discard the domain parameters (cf. Definition 3.2) from the argument list of the algorithms.

3.2.1. Point encoding

Several applications in ECC like point decompression, random sampling of points, and hashing to elliptic curves map to affine points on an elliptic curve. This is what we call point encoding. For a more formal definition we refer to Section 4.3 of [BF03]. In Chapter 7, we present a fault attack on point encoding. Therefore, we now give additional background information on algorithms for point encoding.

Point compression

Let $\mathbb{F}_q$ be a field with characteristic $p > 3$. A point $P \in E(\mathbb{F}_q)$ with $P = (x_P, y_P)$ can be uniquely represented with $\lceil \log_2(q+1) \rceil + 1$ bits as $(x_P, b) \in \mathbb{F}_q \times \{0,1\}$ where $(-1)^b$ is the sign of y_P as defined in Section 2.1.2. This strategy is called *point compression* and the compression function is defined as follows:

$$\begin{aligned} \text{Compress} : \mathbb{F}_q \times \mathbb{F}_q &\to \mathbb{F}_q \times \{0,1\} \\ (x, y) &\mapsto (x, (1 - \text{sign}(y))/2)\,. \end{aligned} \tag{3.4}$$

We see that point compression halves the representation size of points on elliptic curves. Therefore, it is part of many elliptic curve-based schemes and related standards like, e.g., IEEE 1363 [IEE00; IEE13]. The function Compress is injective on $E(\mathbb{F}_q)$ and we denote its inverse, the decompression function, with Decompress. Now, before a compressed point is used as argument of ECSM or of a pairing, it can be completely recovered based on Decompress.

The function Decompress can be computed based on the short Weierstrass equation $y^2 = x^3 + a_4x + a_6$ of E. Algorithm 3.1 lists an example implementation.

Algorithm 3.1 Algorithm Decompress for decompressing points.

Input: $b \in \{0,1\}$, $x, a_4, a_6 \in \mathbb{F}_q$
Output: (x, y) with $y^2 = x^3 + a_4x + a_6$

```
1:  procedure Decompress(x, b)
2:      v ← x^2                                  ▷ v = x^2
3:      v ← v + a_4                              ▷ v = x^2 + a_4
4:      v ← v · x                                ▷ v = x^3 + a_4 x
5:      v ← v + a_6                              ▷ v = x^3 + a_4 x + a_6
6:      if √v ∈ F_q then
7:          v ← √v
8:          y ← (−1)^b v
9:          return (x, y)
10:     else
11:         return O
12:     end if
13: end procedure
```

Similar implementations are proposed in standards [IEE00; IEE13] and used in real-world implementations, for example in Open Secure Socket Layer (OpenSSL). The basic idea of the algorithm is to evaluate the right hand side $x^3 + a_4x + a_6$ of (2.3) for the input (x, b) and then take the square root to find a matching y such that (x, y) is on E. The bit b selects between the two possible square roots $\pm y$. If x is not the x-coordinate of a point in $E(\mathbb{F}_q)$ and hence, $x^3 + a_4x + a_6$ is not a square in $\mathbb{F}_q$, the algorithm returns $\mathcal{O}$.

Algorithm 3.2 Algorithm SamplePoint for sampling uniformly in $E(\mathbb{F}_q)[n]\backslash\{\mathcal{O}\}$.

Input: $E : y^2 = x^3 + a_4x + a_6$, $n \in \mathbb{N}$ with $\#E(\mathbb{F}_q) = cn$
Output: $P \in E(\mathbb{F}_q)[n]\backslash\{\mathcal{O}\}$

```
1: procedure SamplePoint
2:     repeat
3:         Sample (x, b) uniformly at random from F_q × {0,1}.        ▷ sample compression
4:         U ← Decompress(x, b)                                       ▷ decompress to point
5:         P ← cU                                                     ▷ map to subgroup
6:     until P ≠ O
7:     return P
8: end procedure
```

Random point sampling

Based on point decompression, we define an algorithm for sampling points uniformly at random in $E(\mathbb{F}_q)[n]\backslash\{\mathcal{O}\}$ that is shown in Algorithm 3.2. A similar implementation is proposed in IEEE 1363 (cf. Appendix A of [IEE00; IEE13]). The idea of the algorithm is to sample a compressed representation in Line 3, decompress it to a point on the curve in Line 4, and map it to the subgroup with correct order in Line 5. This process is repeated until the compression of a point in $E(\mathbb{F}_q)$ has been sampled and hence, a non-trivial point is returned by Decompress.

Hashing to subgroups of elliptic curves

Some pairing-based protocols require that arbitrary strings are hashed to the domain $\mathcal{G}_1$ or $\mathcal{G}_2$ of the pairing. Examples are the IBE scheme from [BF03] or the short signature scheme from [BLS04b] that we both present in Section 3.3.

We can transform Algorithm 3.2 for random point sampling into an algorithm for hashing to the subgroup $E(\mathbb{F}_q)[n]$ of an elliptic curve. Basically, we replace sampling from $\mathbb{F}_q \times \{0,1\}$ in Line 3 of Algorithm 3.2 by a hash function to $\mathbb{F}_q \times \{0,1\}$. The details of the algorithm are given in Algorithm 3.3 and similar implementations were proposed, for example, in Section 5.2 of [BF03], in Section 3.3 of [BLS04b], or in Section 6.3.3 of the IEEE standard 1363.3 for IBE [IEE13].

As building blocks, we use Algorithm 3.1 and a hash function $\mathrm{H} : \{0,1\}^* \to \mathbb{F}_q \times \{0,1\}$ that can be built based on any standard cryptographic hash function. Based on H, in Line 4, the concatenation of the message M with a fixed length t-bit representation of a counter i is hashed to $(x_P, b) \in \mathbb{F}_q \times \{0,1\}$. Then, in Line 5, the result is decompressed to $E(\mathbb{F}_q)$ and mapped to $E(\mathbb{F}_q)[n]$ in Line 6. We repeat this process up to 2^t times to increase the chance that we hash the input to the compression of a point. Note that with Theorem 2.6, approximately every second element in $\mathbb{F}_q \times \{0,1\}$ is a valid compression. Hence, the probability of failure heuristically decreases exponentially in 2^t.

Algorithm 3.3 Algorithm HashToCurve for hashing to $E(\mathbb{F}_q)[n]$.

Input: $M \in \{0,1\}^*$, $E : y^2 = x^3 + a_4x + a_6$, hash function $\mathrm{H} : \{0,1\}^* \to \mathbb{F}_q \times \{0,1\}$, $n \in \mathbb{N}$ with $\#E(\mathbb{F}_q) = cn$, $t \in \mathbb{N}$
Output: $P \in E(\mathbb{F}_q)[n]$

1: **procedure** HashToCurve(M)
2: $i \leftarrow 0$
3: **repeat**
4: $(x, b) \leftarrow \mathrm{H}\left(M.\lceil i \rceil_2^t\right)$ ▷ hash to $\mathbb{F}_q \times \{0,1\}$
5: $U \leftarrow \mathrm{Decompress}(x, b)$ ▷ decompress to point
6: $P \leftarrow cU$ ▷ map to subgroup
7: $i \leftarrow i + 1$
8: **until** $P \neq \mathcal{O}$ or $i = 2^t$
9: **return** P
10: **end procedure**

3.2.2. Miller algorithm

In Section 2.3, we introduced cryptographic pairings, or more specifically, the (reduced) Tate and the (reduced) Ate pairing. For a given curve E, both pairings are defined based on the so-called Miller function $\mathrm{mil}_{n,P}$ from Definition 2.31 with $n \in \mathbb{Z}$ and $P \in E$. In [Mil04], Victor Miller introduced an efficient algorithm to evaluate $\mathrm{mil}_{n,P}$ at arbitrary points $Q \in E$. The algorithm is therefore called Miller algorithm. In this section, we give more background information on this algorithm because it is the subject of our analysis in Chapter 4, Chapter 5, and Chapter 6. We start with the definition of the algorithm for affine coordinates. Then we explain how it can be computed for Jacobian coordinates.

Affine coordinates

The basic building block of the Miller algorithm is the equation of a line through two given points $P_1, P_2 \in E$ that is defined as follows:

Definition 3.8. *Let $E/\mathbb{K}$ be an elliptic curve given by (2.3). Let $P_1, P_2 \in E\backslash\{\mathcal{O}\}$ with $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$. We define $L_{P_1,P_2}(x, y) \in \overline{\mathbb{K}}(E)$ as follows:*
If $P_1 = -P_2$, define

$$L_{P_1,P_2}(x, y) = x - x_1.$$

If $P_1 \neq -P_2$, let λ_{P_1,P_2} be as in Definition 2.9 and define

$$L_{P_1,P_2}(x, y) = y - y_1 - \lambda_{P_1,P_2}(x - x_1).$$

For the special case where $P_1 = -P_2$, we also write $V_{P_1}(x, y) = L_{P_1,P_2}(x, y)$ and for $P_1 = P_2$, we write $T_{P_1}(x, y) = L_{P_1,P_1}(x, y)$.

Here, $L_{P_1,P_2}(x,y)$ is the line through P_1, P_2, and $-(P_1+P_2)$ as it occurs in the addition of P_1 and P_2. In the case $P_1 = -P_2$, this is the vertical line through P_1 and $-P_1$. In the case $P_1 = P_2$, it is the tangent to E at P_1. We obtain:

Lemma 3.9 (Lemma 1 of [Mil04])**.** *Let P_1, P_2, L_{P_1,P_2}, and V_{P_1} be as in Definition 3.8. Then*

$$\operatorname{div}(L_{P_1,P_2}/V_{P_1+P_2}) = [P_1] + [P_2] - [P_1+P_2] - [\mathcal{O}]\,.$$

The following theorem gives us a recursive approach to construct $\mathrm{mil}_{n,P}$ based on Definition 3.8:

Theorem 3.10. *Let $E/\mathbb{K}$ be an elliptic curve. For $n \in \mathbb{N}$ and $P \in E$ let $\mathrm{mil}_{n,P} \in \overline{\mathbb{K}}(E)$ be as in Definition 2.31. Then it holds that*

$$\mathrm{mil}_{i+j,P} = \mathrm{mil}_{i,P} \cdot \mathrm{mil}_{j,P} \cdot L_{iP,jP}/V_{(i+j)P}. \tag{3.5}$$

Proof. With Definition 2.31 and Lemma 3.9 it follows that both sides have the same divisor. Because $\mathrm{mil}_{n,P}$ is normalized the claim follows from Theorem 2.26. See also Lemma 2 of [Mil04]. □

Now we can construct $\mathrm{mil}_{2i,P}$ from $\mathrm{mil}_{i,P}$ and T_{iP}/V_{2iP} by setting $j = i$ in (3.5) and $\mathrm{mil}_{i+1,P}$ from $\mathrm{mil}_{i,P}$ and $L_{iP,P}/V_{(i+1)P}$ by setting $j = 1$. This allows us to

Algorithm 3.4 Miller algorithm for computing $\mathrm{mil}_{n,P}(Q)$.

Input: $n = \sum_{j=0}^{N-1} n_j 2^j$ with $n_j \in \{0,1\}$ and $n_{N-1} = 1$, $P, Q \in E$
Output: $\mathrm{mil}_{n,P}(Q)$

1: $R \leftarrow P$, $b \leftarrow 1$
2: **for** $j = N-2 \ldots 0$ **do**
3: $\quad b \leftarrow b^2 \cdot T_R(Q)/V_{2R}(Q)$
4: $\quad R \leftarrow 2R$
5: $\quad$ **if** $n_j = 1$ **then**
6: $\quad\quad b \leftarrow b \cdot L_{P,R}(Q)/V_{R+P}(Q)$
7: $\quad\quad R \leftarrow R + P$
8: $\quad$ **end if**
9: **end for**
10: **return** b

efficiently evaluate $\mathrm{mil}_{n,P}$ for arbitrary $n \in \mathbb{N}$ at $Q \in E$ by scanning n from its most significant bit to its least significant bit. The corresponding algorithm is shown in Algorithm 3.4.

Remark 3.11. We can improve the efficiency of Algorithm 3.4 when it is used to compute the reduced Tate or Ate pairing from Definition 2.33 and Definition 2.38, respectively. Assume Type 3 pairing groups (cf. Definition 3.1) with even embedding degree k. It was observed in [BLS04a] that in this case, $V_{2R}(Q)$ in Line 3 and

$V_{R+P}(Q)$ in Line 6 are elements in $\mathbb{F}_{q^{k/2}}$. Hence, they cancel out by exponentiation with $\left(q^k-1\right)/r$ and need not to be computed. This technique is now a standard technique in PBC and called *denominator elimination*.

Jacobian coordinates

Jacobian coordinates are a special form of weighted projective coordinates that are often used in implementations because of their efficiency. We present an attack in Section 4.4 that exploits the representation of R from Algorithm 3.4 in Jacobian coordinates. For a point $(X:Y:Z)\neq\mathcal{O}$ in Jacobian coordinates the corresponding affine representation is given as $(X/Z^2, Y/Z^3)$. For the computation of $P_3=P_1+P_2$ in Jacobian coordinates, we refer to Section 3.2.1 of [Coh+06].

Now, we show how to transform $L_{P_1,P_2}(x,y)$ from Definition 3.8 to Jacobian coordinates. Therefore, let E be given by a short Weierstrass equation (2.3) and let $P_i=(X_i,Y_i,Z_i)$ for $i\in\{1,2\}$. For $P_1\neq P_2$ we define

$$A_{P_1,P_2}=Y_2Z_1^3-Y_1Z_2^3,\quad B_{P_1,P_2}=Z_2^3(X_2Z_1^2-X_1Z_2^2)$$

and for $P_1=P_2$ we define

$$A_{P_1,P_2}=3X_1^2+a_4Z_1^4,\quad B_{P_1,P_2}=2Y_1. \tag{3.6}$$

We obtain the line functions for Jacobian coordinates by clearing denominators after substituting $x_i=X_i/Z_i^2$ and $y_i=Y_i/Z_i^3$ in $L_{P_1,P_2}(x,y)$ from Definition 3.8:

$$L_{P_1,P_2}(x,y)=B_{P_1,P_2}(yZ_1^3-Y_1)-A_{P_1,P_2}(xZ_1^2-X_1). \tag{3.7}$$

In affine coordinates, the computation of λ_{P_1,P_2} from Definition 2.9 requires a costly division. We see from (3.7) that we avoid the division if we represent R from Algorithm 3.4 in Jacobian coordinates.

3.2.3. Final exponentiation

To compute the reduced pairings from Section 2.3.3, the result of Algorithm 3.4 is raised to the power of $\left(q^k-1\right)/r$. This step of the pairing computation is also called the *final exponentiation*. In order to understand our attacks from Chapter 6, we now explain how the exponentiation with $\left(q^k-1\right)/r$ is usually computed. For $d\in\mathbb{N}$, let $\Phi_d(x)$ be the d-th cyclotomic polynomial. It holds that $x^k-1=\prod_{d|k}\Phi_d(x)$. We obtain the following lemma:

Lemma 3.12. *For the field $\mathbb{F}_q$ and a prime r let k be the embedding degree of q and r (cf. Definition 2.16). For $d|k$ it holds that $r|\Phi_d(q)$ if and only if $d=k$.*

Proof. Since $r|q^k-1$ by Definition 2.16 and because $q^k-1=\prod_{d|k}\Phi_d(q)$ it holds that the prime r divides at least one of the factors $\Phi_d(q)$. Assume that $r|\Phi_d(q)$ for $d<k$. Because $\Phi_d(q)|q^d-1$ it follows that $r|q^d-1$. This is a contradiction because the embedding degree k is the smallest integer such that $r|q^k-1$. Hence $r\nmid\Phi_d(q)$ for $d|k$ and $d<k$. Hence, r must divide $\Phi_k(q)$. □

With Lemma 3.12 we can write the exponent of the reduced Tate pairing as

$$\frac{q^k-1}{r} = \frac{q^k-1}{\Phi_k(q)}\frac{\Phi_k(q)}{r}.$$

Based on this factorization, the final exponentiation is typically performed in two steps: In the first step, the exponentiation with the factor $(q^k-1)/\Phi_k(q)$ is computed; and in the second step, the exponentiation with $\Phi_k(q)/r$ is computed. The first factor is light in the sense of Definition 2.1. Hence, exponentiation with this factor can be computed based on efficient applications of the q-th power Frobenius automorphism σ_q and a few multiplications and divisions in $\mathbb{F}_{q^k}$. For the second factor, a standard square-and-multiply approach can be used. For more details and more advanced techniques, we refer to [Sco+09].

Remark 3.13. On the one hand, we explained that the small weight of $(q^k-1)/\Phi_k(q)$ helps us to speed up the final exponentiation. On the other hand, in Chapter 6 we use this property to mount fault attacks on the final exponentiation.

3.3. Pairing-based schemes

In this thesis, we analyze cryptographic schemes with respect to implementation attacks. To motivate our analysis we introduce examples of pairing-based encryption and signature schemes in Section 3.3.1 and Section 3.3.2, respectively. To describe our attacks and attacks of previous work independently from concrete schemes, we introduce three abstract protocols in Section 3.3.3. These protocols subsume applications of real-world schemes like decryption or signature queries that are relevant for our subsequent analysis of vulnerabilities against physical attacks. Vulnerabilities of our abstract protocols do not always transfer to vulnerabilities of the concrete schemes. Nevertheless, our analysis of the abstract protocols helps us to identify critical elements also in concrete implementations.

3.3.1. Pairing-based encryption schemes

In this section, we present examples of pairing-based encryption schemes. First, we present the IBE scheme of Boneh and Franklin [BF03] as an archetype for pairing-based encryption schemes. Then we outline the decryption algorithm of the ABE scheme from Rouselakis and Waters [RW13] to give a more complex application of pairings. Common to both schemes as to most pairing-based encryption schemes is:

1. The secret decryption key is a point on the curve.
2. During decryption, a pairing is computed where one argument of the pairing is the secret key and the other argument is a component of the ciphertext.

This motivates our analysis from Chapter 4, Chapter 5, and Chapter 6 where we analyze the vulnerability of the pairing computation against physical attacks.

Identity-based encryption

We start with the definition of the IBE scheme BasicIdent from [BF03]. As in [CM11], we translate the scheme from the Type 1 setting (see Definition 3.1) with $\mathcal{G}_1 = \mathcal{G}_2$ to the Type 2 or Type 3 setting:

Definition 3.14. *The scheme **BasicIdent** consists of four polynomial time algorithms that are defined as follows:*

1. ***Setup:*** *On input a security parameter λ select $l \in \mathbb{N}$ and domain parameters (cf. Definition 3.2) $D_1 = (\mathbb{F}_q, E_1, R_1, r, c_1)$ and $D_2 = (\mathbb{F}_q, E_2, R_2, r, c_2)$ that define pairing groups $(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T)$ of prime order r. Furthermore select a bilinear map $\mathrm{e} : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$, hash functions $\mathrm{HashToCurve} : \{0,1\}^* \to \mathcal{G}_2$ and $\mathrm{H} : \mathcal{G}_T \to \{0,1\}^l$. Pick s uniformly at random from $\mathbb{Z}/r\mathbb{Z}$ and compute $P_{pub} = sR_1$. Output*

$$pk = (D_1, D_2, \mathrm{e}, \mathrm{HashToCurve}, \mathrm{H}, P_{pub}) \tag{3.8}$$

   as public parameters and s as secret master key.

2. ***Extract:*** *On input an identity $\mathtt{ID} \in \{0,1\}^*$, public parameters pk of the form (3.8), and a secret master key s: compute $P_{\mathtt{ID}} = \mathrm{HashToCurve}(\mathtt{ID})$ and output $\mathtt{ID}$'s private key $Q_{\mathtt{ID}} = sP_{\mathtt{ID}}$.*

3. ***Encrypt:*** *On input a message $M \in \{0,1\}^l$, an identity $\mathtt{ID} \in \{0,1\}^*$, and public parameters pk of the form (3.8) choose $k \in \mathbb{Z}/r\mathbb{Z}$ uniformly at random. Then compute $P_{\mathtt{ID}} = \mathrm{HashToCurve}(\mathtt{ID})$ and*

$$C_1 = kR_1 \qquad\qquad C_2 = M \oplus \mathrm{H}(\mathrm{e}(kP_{pub}, P_{\mathtt{ID}})).$$

   Output $(C_1, C_2) \in E_1 \times \{0,1\}^l$ as the ciphertext.

4. ***Decrypt:*** *On input a ciphertext $(C_1, C_2) \in E_1 \times \{0,1\}^l$, an identity $\mathtt{ID}$, the identity's private key $Q_{\mathtt{ID}}$, and public parameters pk of the form (3.8): compute and output $M = C_2 \oplus \mathrm{H}(\mathrm{e}(C_1, Q_{\mathtt{ID}}))$.*

The Type 1 version of this scheme is IND-CPA secure in the random oracle model if the computational bilinear Diffie-Hellman (CBDH) problem is hard. For the Type 2 and Type 3 versions with $\mathcal{G}_1 \neq \mathcal{G}_2$ the hardness assumptions have to be slightly adapted. For a detailed discussion on this topic, we refer to [CM11]. In [BF03] the Fujisaki-Okamoto transform [FO13] is used to transform BasicIdent into the scheme FullIdent that is IND-CCA secure in the random oracle model. Hence, in the black-box setting, this scheme can be considered as secure. We will see in Chapter 4 and Chapter 6 that this is not necessarily true under physical attacks.

Attribute-based encryption

Pairings cannot only be found in IBE but also in more complicated schemes, e.g., from ABE. For example, consider the key policy ABE scheme from [RW13]. As part of the decryption, a product of the following form is computed:

$$B = \prod_{i \in I} \left(\mathrm{e}(C_0, K_{i,0}) \, \mathrm{e}(C_{\tau(i),1}, K_{i,1}) \, \mathrm{e}(C_{\tau(i),2}, K_{i,2}) \right)^{\omega_i}$$

where the $C_{i,j} \in \mathcal{G}_1$ are components of the ciphertext, the $K_{i,j} \in \mathcal{G}_2$ are components of the recipient's secret key, and the $\omega_i \in \mathbb{Z}$ can be computed from the ciphertext. Then $B \in \mathcal{G}_T$ is used as input of a key derivation function (KDF) or it is used to recover the encrypted message as $M = C/B$ for a ciphertext component $C \in \mathcal{G}_T$. We see that, similar to BasicIdent, the arguments of the pairings are a key component and a component of the ciphertext. Furthermore, the result of the pairing is used to decrypt the message. Different from BasicIdent, not a single pairing, but of a product of pairings is computed.

3.3.2. Pairing-based signature schemes

In this section, we present examples of pairing-based signature schemes. Common to most pairing-based signature schemes is that pairings are used only for signature verification and not for signature generation. Hence, different from pairing-based encryption schemes, the secret key is not an argument of the pairing. Therefore, the pairing itself is less interesting for physical attacks.

We present two types of schemes that differ in the role of the signing key. First, we define the short signature scheme of Boneh, Lynn, and Shacham [BLS04b]. Here, the secret signing key is a scalar that is used as input for ECSM. Then we outline the identity-based signature scheme of Cha and Cheon [CC03]. Here, the secret signing key is a point on the curve. Common to both schemes is that ECSM is computed on the image P of the function HashToCurve defined by Algorithm 3.3. We exploit this property in one of our attacks from Chapter 7.

Signature schemes with secret scalar

The key extraction procedure of an IBE scheme can be transformed into a secure signature algorithm by interpreting the identity `ID` as the message, the master secret key as the signing key, and the identity's private key as signature under message `ID` [BF03]. This is sometimes called the Naor transform and, when applied to the scheme from Definition 3.14, leads us directly to the BLS signature scheme from [BLS04b]:

Definition 3.15. *The **BLS signature scheme** consists of three polynomial time algorithms that are defined as follows:*

1. ***Setup:** On input a security parameter λ select domain parameters (cf. Definition 3.2) $D_1 = (\mathbb{F}_q, E_1, R_1, r, c_1)$ and $D_2 = (\mathbb{F}_q, E_2, R_2, r, c_2)$ that define*

pairing groups $(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T)$ *of prime order* r*. Furthermore select a bilinear map* $\mathrm{e} : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$*, and a hash function* $\mathrm{HashToCurve} : \{0,1\}^* \to \mathcal{G}_1$*. Pick* s *uniformly at random from* $\mathbb{Z}/r\mathbb{Z}$ *and compute* $P_{pub} = sR_2$*. Output*

$$pk = (D_1, D_2, \mathrm{e}, \mathrm{HashToCurve}, P_{pub}) \tag{3.9}$$

as public key and $sk = (pk, s)$ *as secret key.*

2. ***Sign:*** *On input a message* $M \in \{0,1\}^*$ *and a secret key* $sk = (pk, s)$ *compute* $P = \mathrm{HashToCurve}(M) \in \mathcal{G}_1$*,* $Q = sP$*, and output* $\sigma = Q$ *as signature.*

3. ***Verify:*** *On input a message* M*, a signature* $\sigma \in \mathcal{G}_1$*, and a public key* pk *of the form* (3.9)*, output* 1 *if and only if* $\mathrm{e}(\sigma, R_2) = \mathrm{e}(\mathrm{HashToCurve}(M), P_{pub})$*.*

Note that signatures consist of just one $\mathcal{G}_1$ element. For example, let $q = p^k$ and assume the pairing e is instantiated with the reduced Tate pairing $\hat{\mathrm{t}} : E(\mathbb{F}_p)[r] \times E(\mathbb{F}_q)[r] \to \mu_r$ (cf. Definition 2.33). In this case, $\mathcal{G}_1 = E(\mathbb{F}_p)[r]$ and with point compression signatures are only of size $\log p$, i.e., approximately half the size of ECDSA signatures. As it was analyzed in [Cha+10], this scheme is especially efficient if it is instantiated with BN curves from Theorem 3.6. Under the computational Diffie-Hellman (CDH) assumption, the scheme provides existentially unforgeable signatures in a chosen message attack and hence, we consider it as secure in the black-box setting. Again, we will see in Chapter 7 that this is not the case under physical attacks.

The basic structure of the BLS scheme can be found also in other signature schemes with interesting properties. Examples include the threshold signature scheme and the multi-signature scheme from [Bol03], the aggregate signature scheme from [Bon+03], or the signcryption scheme from [LQ04].

Signature scheme with secret point

Another type of pairing-based signature scheme is the identity-based signature scheme from Cha and Cheon [CC03]. Here, and identity $\mathtt{ID}$ has a secret key $D_{\mathtt{ID}}$ on an elliptic curve E. To compute a signature for $\mathtt{ID}$ under a message M we first draw a nonce $s \in \mathbb{Z}/r\mathbb{Z}$ uniformly at random and then compute $P = \mathrm{HashToCurve}(\mathtt{ID})$, $Q = sP$, and $V = (s + \mathrm{H}(M, Q))D_{\mathtt{ID}}$. The signature is the tuple $\sigma = (Q, V)$. We see that the secret is a point on an elliptic curve while ECSM is computed on a nonce s and the image P of HashToCurve.

3.3.3. Abstract protocols

Now we define three abstract protocols between two parties $\mathcal{A}$ and $\mathcal{B}$. These protocols resemble, for example, decryption or signature queries. Later in an attack, $\mathcal{A}$ will take the role of an attacker on the secret key of $\mathcal{B}$. $\mathcal{A}$'s queries model the ability of an adversary to effect $\mathcal{B}$'s computations on the secret key. $\mathcal{B}$'s answers model the information that $\mathcal{A}$ gains during the interaction.

Definition 3.16. *For domain parameters $D = (\mathbb{F}_q, E, R, r, c)$, and $\mathcal{B}$'s secret $s \in \mathbb{Z}/r\mathbb{Z}$ define the protocol* ***Decompress-And-Multiply*** *as follows:*

1. *$\mathcal{A}$ sends a tuple $(x_P, b) \in \mathbb{F}_q \times \{0,1\}$ to $\mathcal{B}$.*
2. *$\mathcal{B}$ computes $P = \text{Decompress}(x_P, b) \in E(\mathbb{F}_q)$ with Algorithm 3.1.*
3. *$\mathcal{B}$ computes $Q = sP$ and returns the result to $\mathcal{A}$.*

This protocol is related to key-agreement and encryption schemes from ECC like Elgamal (cf. Section 20.3 of [Gal12]) when ciphertexts are compressed as described in Section 3.2.1. Here, we can consider a decryption query from $\mathcal{A}$ to $\mathcal{B}$ as an invocation of Decompress-And-Multiply. In Chapter 7, we present an FA on this protocol.

Definition 3.17. *Let $\mathrm{e} : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$ be a pairing. For $i = 1$ let $Q \in \mathcal{G}_2$ be the secret of $\mathcal{B}$ and for $i = 2$ let $P \in \mathcal{G}_1$ be the secret of $\mathcal{B}$. Define the protocol* ***Pair-Argument-****i* *as follows:*

Pair-Argument-1

1. *$\mathcal{A}$ sends $P \in \mathcal{G}_1$ to $\mathcal{B}$.*
2. *$\mathcal{B}$ computes $a = \mathrm{e}(P, Q) \in \mathcal{G}_T$ based on Algorithm 3.4.*
3. *$\mathcal{B}$ returns a to $\mathcal{A}$.*

Pair-Argument-2

1. *$\mathcal{A}$ sends $Q \in \mathcal{G}_2$ to $\mathcal{B}$.*
2. *$\mathcal{B}$ computes $a = \mathrm{e}(P, Q) \in \mathcal{G}_T$ based on Algorithm 3.4.*
3. *$\mathcal{B}$ returns a to $\mathcal{A}$.*

This protocol resembles decryption queries for pairing-based encryption schemes like the IBE and ABE schemes from Section 3.3.1. For example, the connection to the scheme from Definition 3.14 is as follows: Assume that $\mathcal{A}$ queries $\mathcal{B}$ for the decryption of the ciphertext (C_1, C_2). Then $\mathcal{B}$ will compute $\mathrm{e}(C_1, Q_{\mathtt{ID}})$ with his private key $Q_{\mathtt{ID}}$ as part of the decryption. If $\mathcal{A}$ has, for some reason, access to this value, then this can be considered as an invocation of Pair-Argument-1 with $P = C_1$ and $Q = Q_{\mathtt{ID}}$. In Chapter 4 we present SCAs on Pair-Argument while in Chapter 5 and Chapter 6 we present FAs on this protocol.

Definition 3.18. *For domain parameters $D = (\mathbb{F}_q, E, R, r, c)$, and $\mathcal{B}$'s secret $s \in \mathbb{Z}/r\mathbb{Z}$ define the protocol* ***Hash-And-Multiply*** *as follows:*

1. *$\mathcal{A}$ sends a message $M \in \{0,1\}^*$ to $\mathcal{B}$.*
2. *$\mathcal{B}$ computes $P = \text{HashToCurve}(M) \in E(\mathbb{F}_q)$ with Algorithm 3.3.*
3. *$\mathcal{B}$ computes $Q = sP$ and returns the result to $\mathcal{A}$.*

This protocol resembles pairing-based signatures schemes from Section 3.3.2. In the BLS signature scheme from Definition 3.15 for example, signature requests are an invocation of Hash-And-Multiply: Assume $\mathcal{A}$ queries $\mathcal{B}$ for a signature under a

message M. To generate the signature, first, $\mathcal{B}$ hashes $M \in \{0,1\}^*$ to $P \in \mathcal{G}_1$ by means of HashToCurve. The result $P = \text{HashToCurve}(M)$ is used as base-point for the ECSM with the secret key s. Finally, the result of the ECSM is released as the signature $\sigma = Q = sP$. In Chapter 7, we present a FA on this protocol.

Chapter 4.

Side-channel attacks on pairings

In this chapter, we analyze implementations of the protocol Pair-Argument from Definition 3.17 to identify vulnerabilities of pairings against SCAs. Based on our work from [BGL13], we introduce new attacks on the reduced Tate and the reduced Ate pairing from Definition 2.33 and Definition 2.38, respectively. Our attacks use a DPA of the multiplication in finite fields and exploit standard optimizations for efficient implementations of pairings. We remark that we use Pair-Argument as an abstraction of IBE and ABE decryption from Section 3.3.1, but that our results from this chapter directly apply to those concrete applications.

The chapter is structured as follows: In Section 4.1, we give a short introduction into the problem. Then we outline related work and our contribution. In Section 4.2, we present an attack for the Tate pairing in affine coordinates. In Section 4.3, we present an attack for the Ate pairing in affine coordinates. In Section 4.4, we present an attack for the Tate and the Ate pairing in projective coordinates.

4.1. Introduction

A powerful tool in the context of SCAs is a differential power analysis (DPA)[1]. In the setting of a DPA, the attacker targets a binary operation where one operand is part of the secret and the other operand is under control of the attacker. Basically, this allows the attacker to predict the hypothetical output of the operation, based on the knowledge over the controlled operand and on different key hypotheses. Then he correlates the hypotheses with his measurements to find the (partial) key with the highest correlation. To be able to apply a DPA-based attack, the attacker requires some control over one argument of the operation, while the secret has to be computable from the other argument. Now we identify operations in Pair-Argument that are vulnerable to a DPA.

Cryptographic pairings are defined based on finite fields and hence, suitable binary operations for a DPA in the context of Pair-Argument are finite field additions or finite field multiplications. At the heart of Pair-Argument is the computation of a pairing on arguments P and Q with Algorithm 3.4, where one argument is secret

[1] The term correlation power analysis is sometimes understood as a generalization of differential power analysis. We use the term differential power analysis for a statistical analysis, independent of the underlying statistical test, or the concrete physical side-channel (cf. page 161 of [MOP07]).

and the other is under control of an attacker. Looking closer at Algorithm 3.4, we see that the central computations that involve those points are the evaluation of $T_R(Q)/V_{2R}(Q)$ in Line 3 and $L_{P,R}(Q)/V_{R+P}(Q)$ in Line 6. We do not consider $V_{2R}(Q)$ and $V_{R+P}(Q)$ because they are usually not computed due to denominator elimination (see Remark 3.11). Both, $T_R(Q)$ and $L_{P,R}(Q)$, have a similar structure and we concentrate on $T_R(Q)$. We see from Definition 3.8 that with $R = (x_R, y_R)$ and $Q = (x_Q, y_Q)$ the term $T_R(Q)$ expands to

$$T_R(Q) = y_Q - y_R - \lambda_{R,R}\,(x_Q - x_R)\,. \tag{4.1}$$

Here $\lambda_{R,R}$ is the slope of the tangent through R as defined in (2.4) of Definition 2.9. The point R is a multiple of the argument P, and hence R and $\lambda_{R,R}$ depend only on the argument P of the pairing.

From (4.1), we directly identify three possible operations for a DPA that depend on secret and public argument of the pairing: the addition $y_Q - y_R$, the addition $x_Q - x_R$, and the multiplication $\lambda_{R,R}(x_Q - x_R)$. In this thesis, we assume that a multiplication is better suited for a DPA than an addition (see Remark 4.2 later in this section). Therefore, our goal in this chapter is to identify computations that involve modular multiplications on secret and public argument of the pairing and we concentrate on the computation of $\lambda_{R,R}(x_Q - x_R)$.

We see from Definition 3.17 that in an attack on Pair-Argument-1, Q is secret and P and therefore also $\lambda_{R,R}$ are under our control. With knowledge about x_R, we can build hypotheses for the second factor $(x_Q - x_R)$ from hypotheses for x_Q. We are in the setting of a DPA to recover x_Q, and hence also Q.

For Pair-Argument-2, P is secret and Q is under our control. We see that both factors $\lambda_{R,R}$ and $(x_Q - x_R)$ depend on the secret and it is not immediately possible to apply a DPA. Hence, it seams easier to attack Pair-Argument-1 than Pair-Argument-2 with a DPA of the finite field multiplication. Consequently, a DPA for Pair-Argument-1 on the multiplication $\lambda_{R,R}(x_Q - x_R)$ has been described before [WS06]. The authors of [WS06] conclude:

> "We assessed the three candidate pairing algorithms based on the attack on both paths that a secret can take. From this we found that although none of the algorithms assessed proved to be resistant to SCA, Tate and Ate if implemented with the secret being stationed in the first parameter [(Pair-Argument-2)] could withstand such attacks. [...]
>
> From our findings we recommend two straightforward deterrents to protocol designers implementing pairing-based protocols to protect against SCA:
>
> 1. If implementing the Tate or Ate pairing, ensure that the secret parameter is positioned in the first parameter (i.e., the secret takes the P path).
>
> [...]"

We show in this chapter that it is possible to apply a DPA on the modular multiplication $\lambda_{R,R}(x_Q - x_R)$ also for the case Pair-Argument-2 where P is secret. We exploit some standard representations of P and Q that are typically used in efficient implementations. We conclude that one should not rely on inherent countermeasures for secure pairing implementations.

4.1.1. Related work

In [Whe+09] an introduction on SCAs on pairing computations is given. The presentation explains the basic techniques such as DPA and reviews some SCAs on pairings from literature. In the following, we give an extended overview of the related literature.

In [PV04] SCAs were considered for the first time in the context of PBC. Here, the authors identify finite field multiplications in (4.1) that can be used for a DPA. They consider pairings on supersingular curves in characteristic 3. In this case, the term (4.1) has a special form that allows the authors to attack both cases, secret P and secret Q. In addition to the attack, the authors also propose countermeasures based on the randomization of P or Q.

In [WS06] DPA-based attacks on the Tate pairing, the Ate pairing, and the Eta pairing were analyzed. As part of their analysis, the authors of [WS06] describe a concrete DPA of the multiplication in finite fields. Similar to [PV04], they show that the Eta pairing for fields of characteristic 2 is vulnerable to a DPA-based attack for secret P and for secret Q. Furthermore, they show how to attack the multiplication in (4.1) for the Tate and the Ate pairing if the second argument Q represents the secret (Pair-Argument-1). Based on this result, they conjectured that it may be more difficult to attack implementations of the Tate and Ate pairing if the first argument of the pairing is the secret.

In [Kim+06] the vulnerability of the Eta pairing on supersingular curves in characteristic 2 is analyzed. Similar to [WS06], the authors of [Kim+06] conclude that this pairing can be attacked in both cases, secret P and secret Q. They furthermore propose randomization of projective coordinates as a countermeasure.

In [STO08] a new countermeasure is proposed for the Eta pairing in characteristic 3. The authors claim that this countermeasure is more efficient than previous countermeasures such as point blinding or randomized projective coordinates. In [CHK08] the authors improve the efficiency of countermeasures that are based on randomized projective coordinates for the case of pairings over fields of a small characteristic.

In [MFN09] an attack on the Tate pairing is presented for the case where the first argument P is secret (Pair-Argument-2). The authors of [MFN09] assume that the secret is represented in Jacobian coordinates and use a DPA of a modular multiplication *and* a DPA of a modular addition to recover the secret P. Furthermore, they perform a circuit simulation to provide evidence for the feasibility of their attack.

In [PM11] an attack based on a DPA of finite field multiplications is performed.

The authors practically perform the DPA for their field programmable gate array (FPGA) implementation of the Tate pairing on fields of characteristic 2. Hence, they show that the former theoretical attacks are also relevant in practice.

In [GC11] the authors propose a DPA-based attack on the Tate pairing over BN curves. They perform the DPA on an addition in the line computation similar to the addition $(x_Q - x_R)$ in (4.1). The attack is presented for the case where Q is secret. Because the addition $(x_Q - x_R)$ is symmetric in both arguments this is not crucial for the attack. The attack is practically performed on an FPGA-based platform and hence the authors show that a DPA of the modular addition on unprotected implementations is also possible.

In [UW14] a DPA-based attack on the Ate pairing over BN curves is performed. The authors target the multiplication in (4.1). The attack can be applied to both cases, secret P and secret Q. The reason is that BN curves have a twist of degree $d = 6$. This introduces additional structure for P that can be exploited. The authors of [UW14] successfully perform their attack on an FPGA and an ARM Cortex-M0. Hence, they show that attacks based on a DPA of modular multiplications are a serious threat for relevant pairings.

4.1.2. Our contribution

The results that we present in this chapter are mostly based on our results from [BGL13]. There we presented passive attacks on the reduced Tate pairing in affine and in projective coordinates that we recall in this chapter. Additionally we show how our techniques from [BGL13] can be applied to the reduced Ate pairing.

In [WS06] attacks on Tate and Ate pairing were presented for the case of Pair-Argument-1 where the second argument Q is the secret. We extend these attacks to Pair-Argument-2 where P is secret. Furthermore, in [MFN09] and [GC11] attacks on projective coordinates were described that require a DPA of finite field additions. We present our results from [BGL13] that show how the DPA of the addition can be avoided. In some more detail, our contribution is as follows:

1. In Section 4.2 we present our attack from [BGL13] on the reduced Tate pairing in affine coordinates. In this attack, we exploit the structure of $\mathbb{G}_1$ (see Definition 2.18) to transfer the attack of [WS06] from the setting of Pair-Argument-1 to the setting of Pair-Argument-2. Our attack is limited to the case where the embedding degree (cf. Definition 2.16) fulfills $k > 2$. To perform our attack, we exploit that $k > 2$ together with $P \in \mathbb{G}_1$ implies $x(Q) \notin E(\mathbb{F}_q)$. This allows us to separate x_Q and x_R in $\lambda_{R,R}(x_Q - x_R)$ from (4.1). As we explained in Section 3.1.3 the condition $k > 2$ is fulfilled for efficient pairing implementations.

2. In Section 4.3 we present an attack on Pair-Argument-2 when instantiated with the Ate pairing in affine coordinates. In [BGL13] we already exploited the special structure of $\mathbb{G}_2$ (see Definition 2.18) for an attack on the reduced Tate pairing. In Section 4.3 we now apply a similar technique to attack the

Secret	Coordinates	Pairing	k	d	
Q	affine	Tate and Ate	all	all	[WS06]
Q	Jacobian	Tate and Ate	all	all	[MFN09]
P	affine	Tate	> 2	all	Section 4.2 ([BGL13])
P	affine	Ate	$\geq d$	≥ 4	Section 4.3, [UW14]
P	Jacobian	Tate and Ate	all	all	Section 4.4 ([BGL13])

Table 4.1.: Summary of attacks on the reduced Tate and the reduced Ate pairing over large prime fields. Only attacks that are solely based on a DPA of the modular multiplication are considered. Here the columns k and d list the constraints of our attack on the embedding degree and the twist degree, respectively.

reduced Ate pairing that we did not publish so far. For our attack, we require that $\mathbb{G}_2$ is represented based on twists with degree $d \geq 4$ in the form of (3.1) from Section 3.1.2. Again, this allows us to separate x_Q from x_R in $\lambda_{R,R}(x_Q - x_R)$. We remark that the structure of $\mathbb{G}_2$ has also been exploited implicitly in [UW14] for an attack on the Ate pairing over BN curves.

3. In Section 4.4 we present an attack on the reduced Tate and the reduced Ate pairing when computed in (weighted) projective coordinates. For the case of the reduced Tate pairing, we already described the attack in [BGL13]. In Section 4.4 we show that the Ate pairing is also vulnerable to this attack. The attack applies equally to Pair-Argument-1 and Pair-Argument-2 and uses a DPA to recover the z-coordinate of R. Our attack relies on implementations in mixed coordinates [Coh+06, Section 13.2.2] that are typically used for efficiency reasons. This allows us to express the z-coordinate of R as a rational function in P and compute P from this coordinate. Different from the attacks in [MFN09] and [GC11] that require a DPA of the finite field addition in case of secret P, our attack only requires a DPA of the multiplication.

In Table 4.1 we give an overview of our results and related work that target the pairing computation with a DPA of the modular multiplication.

Concerning the impact of our results, today the most efficient implementations use projective coordinates and hence are vulnerable to our attack from Section 4.4. Some results indicate that pairing implementations in affine coordinates become more relevant for modern processors [Aca+13]. These implementations use the Ate pairing and rely on high degree twists. Therefore, they are vulnerable to our attack from Section 4.3. We conclude that SCAs have to be considered for implementations of Pair-Argument-1 *and* Pair-Argument-2 and that countermeasures are required in an environment where implementation attacks are relevant.

Our attacks on Pair-Argument from Definition 3.17 exploit leakage at an internal state of Algorithm 3.4. For the attacks, we require that the pairing is computed on the secret argument and a public argument that is controlled by the attacker.

Because we target an internal state, we do not assume access to the result of the pairing computation. Hence, our attack on Pair-Argument directly implies an attack on IBE or ABE decryption from Section 3.3.1.

4.1.3. Side-channel analysis of modular multiplication

At the beginning of this section, we described a DPA as a tool to solve the problem of recovering one argument of a binary operation in an SCA. Now, we give a formal definition of the problem when the operation in question is a multiplication. The definition adapts the definition of the hidden multiplier problem from [Bel+15] to our application:

Definition 4.1. *Let $a \in \mathbb{Z}$, $\mathcal{B} \subseteq \mathbb{Z}$, and $\sigma \in \mathbb{R}$. Furthermore, let $O_a(\cdot)$ be an oracle that on input $b \in \mathcal{B}$:*

1. *Samples ϵ according to a Gaussian distribution with mean $\mu = 0$ and variance σ.*
2. *Returns $\mathrm{HW}(a \cdot b) + \epsilon$.*

*The **hidden factor problem (HFP)** for $(a, \mathcal{B}, \sigma)$ is to recover a with oracle access to $O_a(\cdot)$.*

The connection to our application is as follows: We assume that elements over a field $\mathbb{F}_q$ with prime q are represented as integers in $[0, q-1]$. Then a in Definition 4.1 represents a fraction of a secret element in $\mathbb{F}_q$ like, for example, one machine word of the secret. Furthermore, $\mathcal{B}$ represents a set of multipliers that we can choose in the attack. The definition of $\mathcal{B}$ depends on the concrete application. For example, it might contain the x-coordinates of points in $E(\mathbb{F}_q)[r]$ that are under the control of an attacker on Pair-Argument. The response $\mathrm{HW}(a \cdot b) + \epsilon$ of the oracle realizes the so-called Hamming weight model [MOP07]. Here, an attacker can observe the Hamming weight of intermediate results disturbed by Gaussian noise with variance σ. Note that the definition can easily be adapted to other leakage models like, for example, the Hamming distance model [MOP07].

Definition 4.1 captures observations that we can make in a practical measurement and allows us to abstract from a concrete side-channel. In our analysis, we try to identify instances of the HFP in Pair-Argument, or in other words, we describe SCAs against Pair-Argument that reduce the recovery of the secret argument of the pairing to instances of the HFP. We do not provide own approaches to solve the HFP itself, e.g., by a DPA. For this matter we refer, for example, to [WS06; UW14; Hut+09; Bel+15] and references therein.

Remark 4.2. So far, we always preferred to target a multiplication and not an addition. To give an intuition for this preference, consider the LSB of the secret argument. On the one hand, in an addition the influence of the LSB on the resulting sum is local. The reason is that the probability of a carry drops exponentially with increasing significance of the bits. On the other hand, for a multiplication the LSB

of one argument effects at least half of the bits of the product. Hence, we assume that it is more difficult to distinguish two key hypotheses with small Hamming distance in a DPA of the addition than in a DPA of the multiplication.

4.2. Analysis of the Tate pairing in affine coordinates

In this section we present an attack on the reduced Tate pairing $\hat{\mathrm{t}}(P,Q)$ from Definition 2.33. In this case, $P \in E(\mathbb{F}_q)$. Furthermore we assume that the embedding degree fulfills $k > 2$.

4.2.1. Identify operands for side-channel

We fix an iteration j of Algorithm 3.4 and analyze $\lambda_{R,R}(x_Q - x_R)$ from (4.1) that is computed in Line 3 of Algorithm 3.4. We exploit the following property in our attack:

Lemma 4.3. *For $E/\mathbb{F}_q$, let r be a prime with $r|\#E(\mathbb{F}_q)$ and $\gcd(r,q) = 1$. Let $Q \in E[r]$ with $Q \notin E(\mathbb{F}_q)$. Furthermore, let $k > 2$ be the embedding degree of q and r. Then it holds that $x(Q) \notin \mathbb{F}_q$.*

Proof. Assume $Q \in E[r]$ with $Q \notin E(\mathbb{F}_q)$ and $x(Q) \in \mathbb{F}_q$. Because the Weierstrass equation (2.3) has a degree of 2 in y it holds that $y(Q) \in \mathbb{F}_{q^2}$. It follows that $Q \in E(\mathbb{F}_{q^2})$. With $Q \in E[r]$ and $Q \notin E(\mathbb{F}_q)[r]$ we obtain a subgroup $\langle Q \rangle \subseteq E(\mathbb{F}_{q^2})[r]$ different from $E(\mathbb{F}_q)[r]$. From Theorem 2.15, it follows that $E[r] \subseteq E(\mathbb{F}_{q^2})$. With Theorem 2.17 this implies $k \leq 2$, a contradiction to $k > 2$. □

Because R is a multiple of P, we obtain $R \in \mathbb{G}_1 \subseteq E(\mathbb{F}_q)$. From the definition of $\lambda_{R,R}$ (cf. Definition 2.9) it follows also that $\lambda_{R,R} \in \mathbb{F}_q$. For a non-degenerate pairing it holds that $Q \notin \mathbb{G}_1$ (cf. Lemma 2.36). Furthermore, from Lemma 4.3 and the fact that $k > 2$, we see that $x_Q \notin \mathbb{F}_q$. Let $\mathbb{F}_{q^k} = \mathbb{F}_q(\theta)$ and write $x_Q = \sum_{i=0}^{k-1} x_Q^{(i)} \theta^i$. If we express the product $\lambda_{R,R}(x_Q - x_R)$ in (4.1) with respect to the basis $\{1, \theta, \ldots, \theta^{k-1}\}$ of $\mathbb{F}_{q^k}$ we obtain

$$\lambda_{R,R}(x_Q - x_R) = \lambda_{R,R}\left(x_Q^{(0)} - x_R\right) + \sum_{i=1}^{k-1} \lambda_{R,R} x_Q^{(i)} \theta^i.$$

Because $x_Q \notin \mathbb{F}_q$ it holds that there exists an $i \in [1, k-1]$ with $x_Q^{(i)} \neq 0$. In Pair-Argument-2, P and hence $\lambda_{R,R}$ is fixed. Furthermore, we are able to control Q and therefore also $x_Q^{(i)}$. We obtain an oracle $O_{\lambda_{R,R}}(\cdot)$ for the HFP from Definition 4.1 that we can query on different elements $b = x_Q^{(i)}$. If we assume an algorithm for solving this HFP instance, we learn $\lambda_{R,R}$.

4.2.2. Recover secret from side-channel information

To recover the secret P from $\lambda_{R,R}$ we proceed in three steps. First, we compute candidate solutions for R. Based on this, we compute candidates for P. Finally, we verify all candidates to identify P.

Compute candidates for R

From (2.4) we obtain

$$\lambda_{R,R} = \frac{3x_R^2 + a_4}{2y_R}.$$

We can write $\lambda_{R,R}$ as a rational function in the coordinates (x_R, y_R) of R. Hence, for fixed $\lambda_{R,R}$ we clear the denominator to obtain the polynomial $f(x,y) = -2y\lambda_{R,R} + 3x^2 + a_4$ with root R. Together with the Weierstrass equation $F(x,y) = y^2 - x^3 - a_4x - a_6$ of E we define the algebraic set $X = V(f(x,y), F(x,y))$. It holds that $R \in X(\mathbb{F}_q)$. Hence, we enumerate $X(\mathbb{F}_q)$ to find candidates for R. This can be done based on appropriate tools like Groebner bases techniques or resultants. We find at most six candidates:

Lemma 4.4. *The algebraic set $X(\mathbb{F}_q)$ contains at most six elements.*

Proof. First note that $F(x,y)$ is irreducible of degree 3. Furthermore, $f(x,y)$ is of degree 2. Hence, $F(x,y)$ and $f(x,y)$ have no common factors. Then it follows from Bezout's Theorem (cf. Theorem 10 in Section 8.§7 of [CLO96]) that $\#X(\mathbb{F}_q) \leq \#X \leq \deg(f)\deg(F) = 6$. □

Compute candidates for P

We know that $R = nP$ for some n that is uniquely determined from Algorithm 3.4, the target iteration j, and the order r of P. Because r is prime, in $\mathbb{Z}/r\mathbb{Z}$ the multiplicative inverse n^{-1} of n exists. Hence, from a candidate $\tilde{R}$ for R we can compute a candidate $\tilde{P} = n^{-1}\tilde{R}$ for P.

Verify candidates

Finally, we can verify each candidate $\tilde{P}$ to identify P. For Pair-Argument-2, where we assume that the attacker has access to the result of the pairing computation, we can obtain $a = \hat{\mathrm{t}}(P,Q)$ for secret P and fixed Q. Then we search for the candidate that fulfills the relation $\hat{\mathrm{t}}(\tilde{P},Q) = a$.

Another way to identify P is to test the functionality of all candidates in the corresponding protocol. For example, in BasicIdent, we could simply check if a candidate point $\tilde{P}$ is a functional decryption key for the attacked identity.

4.2.3. Summary of the analysis

To summarize, we made the reasonable assumption that the embedding degree fulfills $k > 2$. This enabled us to find a component of $x_Q - x_R \in \mathbb{F}_{q^k}$ as a vector in $\mathbb{F}_{q^k}/\mathbb{F}_q$ that is independent from R and hence P. This allowed us to define an oracle for the HFP and to apply an SCA similar to the one presented in [WS06] also for the case where the first argument P is secret.

4.3. Analysis of the Ate pairing in affine coordinates

In this section we present an attack on the reduced Ate pairing from Definition 2.38. In this case, $P \in \mathbb{G}_2 \subseteq E(\mathbb{F}_{q^k})$ and $Q \in \mathbb{G}_1 \subseteq E(\mathbb{F}_q)$. Let E' be a a degree d twist of E with $d \geq 4$, $d|k$, and $\mathbb{F}_{q^k} = \mathbb{F}_{q^{k/d}}(\alpha)$. We assume that ψ is defined as in (3.1) on page 28 and that $\mathbb{G}_2$ is represented based on Theorem 2.21 as $\mathbb{G}_2 = \psi(E'(\mathbb{F}_{q^{k/d}})[r])$. Hence, there is a unique $P' \in E'(\mathbb{F}_{q^{k/d}})$ such that $P = \psi(P')$.

4.3.1. Identify operands for side-channel

Again, we target the multiplication $\lambda_{R,R}\,(x_Q - x_R)$ from (4.1) for a fixed iteration j of Algorithm 3.4. Let $R' \in E'(\mathbb{F}_{q^{k/d}})$ with $R' = (x_{R'}, y_{R'})$ and $R = \psi(R') = (\alpha^2 x_{R'}, \alpha^3 y_{R'})$. Furthermore, let $\lambda_{R',R'} \in E'(\mathbb{F}_{q^{k/d}})$ be the slope of the tangent $T_{R'}(x, y)$ to E' at R'. Let $y^2 = x^3 + a_4 x + a_6$ and $y^2 = x^3 + a'_4 x + a'_6$ be the defining equations of E and E', respectively. With (3.2) from page 28 it holds that $a_4 = \alpha^4 a'_4$ and hence

$$\lambda_{R,R} = \frac{3\left(\alpha^2 x_{R'}\right)^2 + \alpha^4 a'_4}{2\left(\alpha^3 y_{R'}\right)} = \alpha \lambda_{R',R'}.$$

We obtain

$$\lambda_{R,R}\,(x_Q - x_R) = \alpha \lambda_{R',R'}\left(x_Q - \alpha^2 x_{R'}\right). \tag{4.2}$$

For $d \geq 4$ we sort terms in (4.2) with respect to the basis $\{1, \alpha, \ldots, \alpha^{d-1}\}$ of $\mathbb{F}_{q^k} = \mathbb{F}_{q^{k/d}}(\alpha)$:

$$\lambda_{R,R}\,(x_Q - x_R) = \lambda_{R',R'} x_Q\,\alpha - \lambda_{R',R'} x_{R'}\,\alpha^3.$$

Hence, we successfully separated x_Q from $x_{R'}$.

With $\lambda_{R',R'} \in \mathbb{F}_{q^{k/d}}$ and $x_Q \in \mathbb{F}_q$ the multiplication $\lambda_{R',R'} \cdot x_Q$ involves k/d multiplications in $\mathbb{F}_q$. We consider $\mathbb{F}_{q^{k/d}}/\mathbb{F}_q$ as a $\mathbb{F}_q$ vector space with basis $\{1, \theta, \ldots, \theta^{k/d-1}\}$. With $\lambda_{R',R'} = \sum_{i=0}^{k/d-1} \lambda^{(i)}_{R',R'} \theta^i$ we write

$$\lambda_{R',R'} x_Q = \sum_{i=0}^{k/d-1} \lambda^{(i)}_{R',R'} x_Q\,\theta^i.$$

In Pair-Argument-2 the first argument and hence $\lambda_{R',R'}$ is fixed while the second argument Q is under our control. For all $i \in [0, k/d-1]$ we obtain an oracle

$O_{\lambda^{(i)}_{R',R'}}(\cdot)$ for the HFP from Definition 4.1 that we can query on different elements $b = x_Q$. If we assume an algorithm for solving this HFP instance, we learn $\lambda^{(i)}_{R',R'}$. We repeat the analysis for k/d times to recover $\lambda^{(0)}_{R',R'}, \cdots, \lambda^{(k/d-1)}_{R',R'}$ and hence the complete $\lambda_{R',R'}$.

Remark 4.5. Note that with a twist of degree $d = 2$ this attack is not possible. To see this, let $\alpha^2 \equiv a_0 + a_1\alpha$ in $\mathbb{F}_{q^k} = \mathbb{F}_{q^{k/d}}(\alpha)$. For (4.2) we now obtain

$$\alpha\lambda_{R',R'}\left(x_Q - \alpha^2 x_{R'}\right) = \alpha\lambda_{R',R'}\left(x_Q - a_0 x_{R'} - a_1 x_{R'}\alpha\right).$$

Hence, x_Q and $-a_0 x_{R'}$ are added in $\mathbb{F}_q$ and we are not able to separate x_Q from $x_{R'}$ as required for the attack.

4.3.2. Recover secret from side-channel information

To recover the secret P' from $\lambda_{R',R'}$ we proceed exactly as in Section 4.2 with the only difference that we perform all computations on $E'(\mathbb{F}_{q^{k/d}})$ and not on $E(\mathbb{F}_q)$.

First, we use the equation of $\lambda_{R',R'}$ to define $f(x, y) = -2y\lambda_{R',R'} + 3x^2 + a'_4$ with root R'. Then we use the Weierstrass equation $F(x, y) = y^2 - x^3 - a'_4 x - a'_6$ of E' to define $X = V(f(x, y), F(x, y))$ with $R' \in X(\mathbb{F}_{q^{k/d}})$. We enumerate $X(\mathbb{F}_{q^{k/d}})$ with appropriate tools and obtain at most six candidates for R'. From candidates for R' we compute and verify candidates for P' as in Section 4.2.

We remark that the enumeration of $X(\mathbb{F}_{q^{k/d}})$ is more complex than it has been for the Tate pairing because we work over the larger field $\mathbb{F}_{q^{k/d}}$.

4.3.3. Summary of the analysis

To summarize the attack: We made the assumption that E' admits a twist of degree $d \geq 4$. Furthermore we assumed that the target implementation uses the sparse representation from (3.3) for elements in $\mathbb{G}_2$. As in Section 4.2 this enabled us to find a component of $x_Q - x_R \in \mathbb{F}_{q^k}$ as a vector in $\mathbb{F}_{q^k}/\mathbb{F}_{q^{k/d}}$ that is independent from R and hence P. Based on this we were able to define an oracle for the HFP and apply an SCA similar to the one presented in [WS06] also for the case where the first argument P is secret.

In [BGL13] we already used the sparse representation of $\mathbb{G}_2$ to present an attack on the Tate pairing. More specifically, we used the sparseness of (4.2) to target either the multiplication of $b = T_P(Q)$ and $L_{2P,P}(Q)$ in Line 6 during the first iteration of Algorithm 3.4 or the squaring of $b = T_P(Q)$ in Line 3 during the second iteration of Algorithm 3.4.

We remark that in the attack on the Ate pairing from [UW14], the sparseness of (4.2) was exploited for the special case of BN curves (cf. Section 3.1.3) that have a twist of degree $d = 6$.

4.4. Analysis of pairings in projective coordinates

In this section, we present an attack on the reduced Tate and the reduced Ate pairing. This time, we assume that the variable point R of Algorithm 3.4 is represented in Jacobian coordinates. Furthermore, we assume that the arguments P and Q are still given in affine coordinates, or equivalently, that their z-coordinates are set to 1. These so-called mixed coordinates are typically used in efficient implementations (cf. Section 13.2.2 of [Coh+06]).

Here, we present the attack for Jacobian projective coordinates that we introduced in Section 3.2.2. We remark that the same attack applies to other forms of (weighted) projective coordinates.

4.4.1. Identify operands for side-channel

Again, we fix an iteration j of Algorithm 3.4 and analyze the tangent line $T_R(Q)$ from (4.1). Let $R = (X_R, Y_R, Z_R)$ be a fixed representative of R in Jacobian coordinates. From (3.6) and (3.7) we obtain the following representation of (4.1) for Jacobian coordinates:

$$T_R(Q) = 2Y_R \left(y_Q Z_R^3 - Y_R\right) - \left(3X_R^2 + a_4 Z_R^4\right) \left(x_Q Z_R^2 - X_R\right). \tag{4.3}$$

There are two obvious operations that lead to instances of the HFP: the multiplication of y_Q with Z_R^3 and the multiplication of x_Q with Z_R^2. The outline of the attack is similar for both cases and we continue with the computation of $x_Q Z_R^2$.

Reduced Tate pairing

For the case of the reduced Tate pairing it holds that $Z_R \in \mathbb{F}_q$ and $x_Q \in \mathbb{F}_{q^k}$. For $\mathbb{F}_{q^k} = \mathbb{F}_q(\theta)$, we express x_Q with respect to the basis $\{1, \theta, \ldots, \theta^{k-1}\}$ as $x_Q = \sum_{i=0}^{k} x_Q^{(i)} \theta^i$. Then, we expand the product $x_Q Z_R^2$:

$$x_Q Z_R^2 = \sum_{i=0}^{k-1} x_Q^{(i)} Z_R^2 \, \theta^i.$$

In Pair-Argument-2 we can control Q, and hence also $x_Q^{(i)}$ for $i \in [0, k-1]$ while P and therefore also Z_R is fixed. We obtain an oracle $O_{Z_R^2}(\,\cdot\,)$ for the HFP from Definition 4.1 that we can query on different elements $b = x_Q^{(i)}$ for an $i \in [0, \ldots, k-1]$. If we assume an algorithm for solving this HFP instance, we learn Z_R^2.

Reduced Ate pairing

For the case of the reduced Ate pairing, it holds that $x_Q \in \mathbb{F}_q$ and $Z_R \in \mathbb{F}_{q^k}$. This time, we express Z_R^2 as $Z_R^2 = \sum_{i=0}^{k-1} Z_R^{(i)} \theta^i$. We expand the product $x_Q Z_R^2$:

$$x_Q Z_R^2 = \sum_{i=0}^{k-1} x_Q Z_R^{(i)} \, \theta^i.$$

For each $i \in [0, k-1]$ we obtain an oracle $O_{Z_R^{(i)}}(\cdot)$ for the HFP from Definition 4.1. We can can query the oracle on different elements $b = x_Q$. If we assume an algorithm for solving these HFP instances, we learn $Z_R^{(i)}$ for each $i \in [0, k-1]$. This allows us to recover the complete Z_R^2.

Remark 4.6. If we make the same assumption as in Section 4.3 that $\mathbb{G}_2$ is represented based on a degree d twist E' of E, then we can write $R = \psi(R')$ for $R' \in E'(\mathbb{F}_{q^{k/d}})$ with ψ from Definition 2.13. Then $Z_R \in \mathbb{F}_{q^{k/d}}$ and only k/d instances of the HFP have to be solved.

In Section 4.3 this efficient representation of $\mathbb{G}_2$ was crucial in order to mount the attack. Here, this representation reduces the number of HFP instances by a factor of d but is not necessarily required.

4.4.2. Recover secret from side-channel information

The point R in iteration j of Algorithm 3.4 is a fixed multiple of the point P. Let $R = nP$ for some $n \in \mathbb{N}$. Now we express the multiplication-by-n map $[n] : E \to E$ as a rational map of projective algebraic sets of the form $[n] = (\psi_X, \psi_Y, \psi_Z)$. We assume that we know the implementation of Algorithm 3.4, and especially the concrete implementation of the group law in projective coordinates. We recursively apply this implementation to variables (x, y, z) to explicitly construct ψ_X, ψ_Y, and ψ_Z. Now we use the assumption that P is normalized with z-coordinate 1. We de-homogenize $[n]$ by setting $z = 1$ to obtain $\psi_X, \psi_Y, \psi_Z \in \mathbb{F}_q[x, y]$ and $\psi_Z(P) = Z_R$. Hence P is a root of the polynomial $f(x, y) = \psi_Z(x, y)^2 - Z_R^2$.

With the Weierstrass equation $F(x, y) = y^2 - x^3 - a_4 x - a_6$ we define the algebraic set $X = V(f(x, y), F(x, y))$. Then $P \in X(\mathbb{F}_{q^k})$. Hence, we can find P by first enumerating $X(\mathbb{F}_{q^k})$ and then verifying candidates as in Section 4.2. Note that for the Tate pairing with $P \in E(\mathbb{F}_q)$ we can restrict our search to $X(\mathbb{F}_q)$ for an improved efficiency.

From the addition and doubling formulas for projective coordinates (see, for example, Section 13.2 of [Coh+06]), we see that n and the degree of $f(x, y)$ grow exponentially in the attacked iteration j. Hence, we have to apply the attack for small j. More precisely, for Jacobian coordinates $\psi_Z(x, y)$ is well known. Here, it holds that $\psi_Z(x, y) = c\psi_n(x, y)$ where ψ_n is the n-th division polynomial (see Section 3.2 of [Was03]) and $c \in \mathbb{F}_{q^k}$ is a constant that depends on the concrete implementation of the group law. On E, $\psi_n(x, y)^2$ and hence $f(x, y)$ can be written as a univariate polynomial in x of degree $n^2 - 1$ [Was03, Lemma 3.5].

Example 4.7. For a small example, assume that we target the second iteration of Algorithm 3.4 and that no addition in Line 7 occurs in the first iteration. Then $j = 2$, $n = 2$, and $R = 2P$. From the group law in Jacobian coordinates it follows that $Z_R = 2y_P$. We obtain $f(x, y) = 4y_P^2 - Z_R^2$ of degree 2. On E, we can substitute $y_P^2 = x_P^3 + a_4 x_P + a_6$ to obtain a univariate polynomial of degree $n^2 - 1 = 3$.

We performed the analysis for randomly chosen $P \in E(\mathbb{F}_q)[r]$ with a Pari/GP [PAR12] based implementation. We computed $X(\mathbb{F}_q)$ by factoring the resultant of

$f(x, y)$ and $F(x, y)$ over $\mathbb{F}_q$ to obtain x-coordinates of $X(\mathbb{F}_q)$ that we extended to y-coordinates based on the Weierstrass equation $F(x, y)$. We were able to recover P from Z_R^2 for $j \leq 5$ and fields of size $\log(q) = 1024$ in a few minutes.

Remark 4.8. The recovery of P from Z_R is exactly where our our analysis differs from [MFN09]. In [MFN09], with knowledge of Z_R^2, a second DPA on the modular addition of $x_Q Z_R^2 - X_R$ in (4.3) is used to recover X_R. From Z_R and X_R, the point R can be recovered based on the Weierstrass equation in Jacobian coordinates. Then P can be computed as $P = n^{-1}R$. We avoid the additional DPA of the addition with our algebraic approach.

4.4.3. Summary of the analysis

In summary, we described how to apply the HFP to recover the z-coordinate of an intermediate point. We assumed a normalized argument $P = (x_P, y_p, 1)$ of the pairing. This assumption is reasonable because it offers more efficient implementations. We showed that under this assumption it is possible to recover the secret point P directly from the obtained z-coordinate if one of the early iterations of Algorithm 3.4 is attacked.

With respect to the recovery of the z-coordinate, our attack is similar to the attack of [MFN09]. But while [MFN09] require a DPA of a modular multiplication *and* a modular addition to obtain P, our algebraic approach avoids the DPA of the addition.

Chapter 5.

Framework for the analysis of fault attacks on pairings

In this chapter, we turn from passive attacks to active attacks. We introduce a framework for the systematic analysis of fault attacks on pairings, or more specifically, on the protocol Pair-Argument from Definition 3.17. Although several fault attacks on the pairing computation were described in the literature, to the best of our knowledge, no systematization of fault analysis of pairings exists. In this chapter, we close this gap and provide a unified framework that helps to estimate the efficiency of current and future attacks and that helps to automate the analysis of new attacks. We already took the first steps into this direction in the analysis of our fault attack on pairings in [Blö+14], but in this generality we did not publish this framework so far.

The outline of this chapter is as follows: In Section 5.1 we give background information on pairing inversion, we present related work, and we outline our contribution. In Section 5.2 we provide our framework for the analysis of fault attacks on pairings. In Section 5.3 we provide additional background information on the tools of the framework. In Section 5.4 we apply the framework to selected attacks from the literature.

5.1. Introduction

We see from Definition 3.17 that a key recovery attack without faults on Pair-Argument has to invert the pairing in one of its arguments. In order to understand how faults can facilitate this task, we give some background information on pairing inversion. In Section 5.1.1 we start with a formal definition of pairing inversion and outline the basic algebraic approach to the problem. We will conclude that in general, second order faults are required for an attack. Then, in Section 5.1.2 we introduce the hidden root problem (HRP) from [Ver08] that connects pairing inversion with fault attacks on pairings. The HRP and the techniques of [Ver08] to approach the HRP are an important part of our framework. In Section 5.1.3 and Section 5.1.4 we summarize related work and our contribution, respectively. In Section 5.1.5 we shortly explain how FAs on Pair-Argument relate to FAs on concrete pairing-based schemes.

5.1.1. Pairing inversion

The problem of pairing inversion has been studied in the literature, for example in [Sat07; GHV08]. Typically, pairing inversion is defined as follows (cf. Definition 1 of [GHV08]):

Definition 5.1. *Let $i \in \{1,2\}$, $(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T)$ be pairing groups, and $\mathrm{e} : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$ be a bilinear pairing. Furthermore, let $P_i \in \mathcal{G}_i$ be fixed. Given P_i and $a \in \mathcal{G}_T$, the **fixed argument pairing inversion (FAPI)-i problem** is to compute $P_{3-i} \in \mathcal{G}_{3-i}$ such that $a = \mathrm{e}(P_1, P_2)$.*

We see that FAPI-i corresponds to the analysis of Pair-Argument-i for $i \in \{1,2\}$. For pairings that are useful in cryptography, the problem of FAPI can be considered as hard. For example, if both the FAPI-1 problem and the FAPI-2 problem can be solved efficiently for $(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T)$, then also the CDH problem can be solved efficiently in $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_T$ (cf. Section III of [GHV08]).

Now we outline an approach to solve FAPI because it leads us to the analysis of fault attacks on pairings. To make it more explicit, we give our description for the reduced Tate pairing from Definition 2.33. For now, assume an attack on Pair-Argument-1: i.e., that the second argument of the pairing is secret. For the groups $\mathbb{G}_1, \mathbb{G}_2 \subseteq E(\mathbb{F}_{q^k})[r]$ (cf. Definition 2.18) let $a = \mathrm{mil}_{r,P}(Q)^{\left(q^k-1\right)/r}$ be the result of the reduced Tate pairing on input $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$. We can now consider pairing inversion as a two-step process consisting of exponentiation inversion (EI) and Miller inversion (MI) [GHV08]:

1. Exponentiation inversion: Compute the $\left(q^k-1\right)/r$-th roots of a to find b in the image of $\mathrm{mil}_{r,P}(x,y)$ on the domain $\mathbb{G}_2$.

2. Miller inversion: Compute solutions of $b = \mathrm{mil}_{r,P}(x,y)$ for fixed b to find Q.

It is shown in Section VII.D of [GHV08] that the degree of non-degenerate pairings is large, at least in the order of r. Hence, at least one of $\left(q^k-1\right)/r$ or $\deg(\mathrm{mil}_{r,P})$ is large. If $\left(q^k-1\right)/r$ is large, EI is difficult because we do not know how to efficiently identify the $\left(q^k-1\right)/r$-th root b of a that fulfills $\mathrm{mil}_{r,P}(Q) = b$. If $\deg(\mathrm{mil}_{r,P})$ is large, for MI, we need to solve a polynomial equation of large degree. In general, this is also difficult.

We conclude that in general, two faults are required to attack a pairing. With the first fault, we reduce the degree of the Miller function $\mathrm{mil}_{r,P}(x,y)$ to facilitate MI. With the second fault, we introduce an algebraic structure into the final exponent to facilitate EI.

5.1.2. The hidden root problem

Assume a fault is introduced into the computation of $\mathrm{mil}_{r,P}(Q)$ such that its secret argument can be described as the root of a small-degree polynomial. In this case we are left with EI. In [Ver08], Vercauteren studies this problem. We start with

the definition of the (subfield) HRP that is basically an adaption of the definition from [Ver08] to our notation:

Definition 5.2. *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be algebraic sets defined over $\mathbb{F}_{q^k}$. Let $e \in \mathbb{N}$ with $e|q^k - 1$ and $\{f_x : \mathcal{D}_2 \to \mathbb{F}_{q^k}\}_{x \in \mathcal{D}_1}$ be a family of keyed maps. Let $S \in \mathcal{D}_1$ be fixed. The **hidden root problem** is the problem to recover S with oracle access to $f_S(\cdot)^e$. If $\mathcal{D}_1$ is defined already over a proper subfield $\mathbb{F}_{q^d}$ of $\mathbb{F}_{q^k}$, we call the HRP also **subfield** HRP.*

We demonstrate the connection of the HRP to fault attacks on PBC in an example.

Example 5.3. Consider the protocol Pair-Argument-2 from Definition 3.17 and let the pairing be instantiated with the reduced Tate pairing from Definition 2.33 with $\mathcal{G}_1 = E(\mathbb{F}_q)$ and $\mathcal{G}_2 = \mathbb{G}_2 \subseteq E(\mathbb{F}_{q^k})$. Let $S \in \mathcal{G}_1$ be the secret key of $\mathcal{B}$. Furthermore, assume an attacker $\mathcal{A}$ is allowed to query $\mathcal{B}$ for the computation of the pairing on elements $Q \in \mathcal{G}_2$. Then $\mathcal{A}$ introduces a fault at Algorithm 3.4 such that $T_S(Q)^e$ from Definition 3.8 is computed in place of $\mathrm{mil}_{r,S}(Q)^e$. This gives $\mathcal{A}$ an oracle for the function $T_S(\cdot)^e$. With $\mathcal{D}_i = \mathcal{G}_i$ and $S \in \mathcal{D}_1$, we are in the setting of Definition 5.2. Even more specifically, because S is already defined over $\mathbb{F}_q \subset \mathbb{F}_{q^k}$, this is an instance of the subfield HRP.

Vercauteren [Ver08] gives an approach for solving the HRP. A crucial step in the analysis is, for a given exponent e, to find an integer $u < (q^k - 1)/e$ such that $ue = \sum_{i \leq 0} \epsilon_i q^i$ has a small weight $\mathrm{w}_q(ue)$ (cf. Definition 2.1).

Let $\Phi_k(x)$ be the k-th cyclotomic polynomial. Vercauteren concludes that the approach is not efficient for final exponents of the form $e = \left(q^k - 1\right)/r$ where r is a proper divisor of $\Phi_k(q)$ with a large cofactor. For the reduced Tate and reduced Ate pairing, this is typically the case:

Lemma 5.4. *For $E/\mathbb{F}_q$ with $r|\#E(\mathbb{F}_q)$ let $2 < k < 105$ be the embedding degree of $q > 2$ and r. Then $\Phi_k(q)/r \geq q/4$.*

Proof. For $k < 105$, the coefficients of $\Phi_k(x)$ are in $\{0, -1, 1\}$ with leading coefficient 1 [Lan02]. With $n = \deg(\Phi_k(x))$ it follows that $\Phi_k(q) \geq q^n - \sum_{i=0}^{n-1} q^i$. For $q > 2$ it holds that $\sum_{i=0}^{n-1} q^i \leq q^n/2$ and hence $\Phi_k(q) \geq q^n/2$. From Lemma 3.12 we see that $r|\Phi_k(q)$. Because $r|\#E(\mathbb{F}_q)$ it follows from Theorem 2.6 that $r \leq 2q$ and hence $\Phi_k(q)/r \geq (q^n/2)/(2q)$. For $k > 2$ it holds that $n \geq 2$ and the claim follows. □

Hence, for fault attacks on the reduced Tate and the reduced Ate pairing, the approach of [Ver08] is efficient only if also faults at the final exponentiation are considered. In this chapter, we extend the approach of [Ver08] into this direction. We also give more background information on the individual steps from above.

5.1.3. Related work

Several fault attacks on pairings use a fault to facilitate MI: e.g., [PV06; WS07; Mra10; Mra09; BMH13] and in Section 6.1.1 we give a more detailed overview of existing attacks. All those attacks reduce the degree of $\mathrm{mil}_{r,P}(Q)$ as a function in the secret argument by means of faults. In the case where the final exponent already has a special structure, one fault is sufficient. Indeed, as we will see in Section 6.1.1, all fault attacks on a complete pairing computation that use only one fault assume a special structure of the final exponent [PV06; WS07]. Only a few attacks target the final exponentiation to facilitate EI. Exceptions are [LFG13] and our attacks from Chapter 6. Even though several theoretical fault attacks on pairings were published, so far no systematization of the analysis exists. The only work in this direction is [Ver08] which studies the HRP and provides a systematic approach to solve it.

Because the analysis of fault attacks on pairings is related to pairing inversion, we now give some examples from literature that study FAPI. In [GHV08] Galbraith, Hess, and Vercauteren give an extensive overview of pairing inversion. They relate the hardness of pairing inversion to problems like the CDH problem and the bilinear Diffie-Hellman (BDH) problem. Furthermore, it was already observed in [GHV08] that some pairings have a Miller function of polynomial degree such that MI can be solved algebraically in polynomial time. The authors of [GHV08] remark that this approach quickly becomes inefficient for growing degrees of the Miller function and that it might be faster to solve FAPI by computing the DLOG in $\mathbb{F}_{q^k}^*$. Furthermore, in [GhS07] and [Sat13], FAPI is studied for the case where the exponent has a special algebraic structure that allows to reduce FAPI to MI.

There is a line of work, including for example [KO12] and [Cha+14], that reduces FAPI of more general pairings to EI of so-called auxiliary pairings with a small degree Miller function. For this reduction, an oracle for EI on the auxiliary pairing is required in order to access the output of the auxiliary Miller function. Galbraith argues in [Gal13] that EI cannot be well-defined without reference to the solution of pairing inversion and he doubts that this approach leads to interesting insights to pairing inversion.

5.1.4. Our contribution

Based on [Ver08], we describe a general framework for the evaluation of FAs on pairings with the goal to provide a unified approach for analyzing previous and future attacks. Our framework considers faults in both steps, Miller algorithm and final exponentiation, of the pairing computation. Our main contribution of this chapter is to present the techniques from [Ver08] with focus on fault attacks on pairings. Furthermore, we extend the work of [Ver08] in some details:

- In Section 5.2.1 we define a formal model for fault attacks on pairings that, additionally to [Ver08], also covers error terms in $\mathbb{F}_q$ that are introduced at fault injection. Based on our definition, we are able to give a unified analysis

for either case where the first argument P or the second argument Q of the pairing is secret.

- In Section 5.3.2 we provide prototypes that show how to express relevant attacks from literature with our model.
- In Section 5.3.3 and Section 5.3.5 we give more details on how faults at the final exponentiation can be handled by the analysis.
- The analysis of [Ver08] uses Weil restriction of scalars from Definition 2.2. In Section 5.3.6 we give a detailed description of how Weil restriction of scalars applies to the analysis.
- In Section 5.4 we apply our framework to the attacks from [PV06] and [LFG13]. The goal is to demonstrate the individual steps of our analysis. Furthermore, the systematic analysis of those attacks highlights their basic ideas.

We remark that our contribution of this chapter has not been published so far.

5.1.5. Application to concrete schemes

In the abstract protocol Pair-Argument, $\mathcal{A}$ has access to the result $a = \mathrm{e}(P, Q)$ of the pairing computation. For our attacks, also as for previous attacks on pairings, access to a is required. One the one hand, this shows that the result of the pairing computation has to be protected carefully to prevent attacks. On the other hand, in [CKM15] the authors argue that in most pairing-based schemes, the adversary cannot directly access the result of the pairing computation. The reason is that in a practical system like BasicIdent from Definition 3.14, a KDF or a hash function is applied to map $\mathrm{e}(P, Q)$ to $\{0, 1\}^*$. Hence, to turn an attack on Pair-Argument into an attack on a real scheme, the KDF has to be inverted by other means, for example, by an additional fault.

5.2. Description of the framework

In this section, we define our framework for the analysis of FAs on pairings. In Section 5.2.1 we give the necessary definitions and in Section 5.2.2 we outline the steps of the analysis.

5.2.1. Models for fault attacks on pairings

Now we introduce two models. We use the first model to describe an attack. With the second model, we describe the input of the mathematical analysis.

Model for the attack

Inspired by the HRP [Ver08] from Definition 5.2, we model fault attacks on pairing computations based on an oracle that allows us to evaluate a set of functions that depend on the targeted secret:

Definition 5.5. *Let $\mathcal{D} \subseteq \mathbb{A}^n$ be an algebraic set defined over $\mathbb{F}_q$. We say that an attack implements an oracle for a set $\mathcal{M} \subseteq \overline{\mathbb{F}}_q(x_1, \ldots, x_n)$ of rational functions, if, for all $f \in \mathcal{M}$, the attack allows us to evaluate f on the elements of $\mathcal{D}$.*

The idea is that $\mathcal{M}$ models a set of functions that an attacker can realize in an attack by modifying the pairing computation by means of faults. As in Definition 5.2 these functions will depend on the secret S. With oracle access to $\mathcal{M}$, it will be the attacker's task to recover the secret S. Here, $\mathcal{D}$ represents the domain of the pairing's public argument. For example, the attack from Example 5.3 implements an oracle for the set $\mathcal{M} = \{T_S(x, y)^e\}$ on the domain $\mathcal{D} = \mathbb{G}_2 \subseteq E(\mathbb{F}_{q^k})$. Furthermore, $\mathcal{D}$ also models faults in $\mathbb{F}_q$ that are introduced by the attack. Hence, depending on the fault model and different from Definition 5.2, the attacker has not necessarily full control over the arguments of the functions in $\mathcal{M}$. We provide examples later in Section 5.3.2 and Section 5.4.2.

Model for the secret

Now we give a definition that describes the secret argument S of the pairing by a set of rational functions. This is what we call a model for a secret S:

Definition 5.6. *Let $S \in E(\mathbb{F}_q)$ and $\mathcal{F} \subset \mathbb{F}_q(x_1, \ldots, x_n)$. We say that $\mathcal{F}$ is an $\mathbb{F}_q$-**rational model** for S if there exists $(\alpha_1, \ldots, \alpha_n) \in \mathbb{A}^n(\mathbb{F}_q)$ with the following properties:*

1. *For all $f \in \mathcal{F}$ it holds that $f(\alpha_1, \ldots, \alpha_n) = 0$.*
2. *S can be efficiently computed from $(\alpha_1, \ldots, \alpha_n)$.*

If, additionally, the degree of every element in $\mathcal{F}$ is bounded by d, then we say that $\mathcal{F}$ is a degree-d model for S.

We see that a model for a secret S encodes S as a solution of a system of rational equations. Furthermore, to be a solution of the model is a necessary and not a sufficient condition for encoding S. Hence, in general, the model does not uniquely determine S.

In an attack that implements an oracle for a set $\mathcal{M}$ of rational functions in the sense of Definition 5.5, there is a canonical way to deduce a model for a secret $S \in E$:

1. Let $f_S \in \mathcal{M}$ and $a = f_S(Q)$ be the response of the oracle on input Q.
2. With $P = (x, y)$, add $a - f_P(Q) \subseteq \mathbb{F}_q(x, y)$ to the model.

3. Add the Weierstrass equation (2.3) that defines E to the model.

For the attack from Example 5.3 we obtain $\mathcal{M} = \{T_S(\,\cdot\,)^e\}$ and $a = T_S(Q)^e$. Let $P = (x, y)$. Then, $f(x, y) = a - T_P(Q)^e$ together with the Weierstrass equation of the curve define a model for S. As we will see in Section 5.3.2, there are examples where the canonical approach fails because the resulting initial model has too large degree.

Remark on notation

The descriptions of oracle and model depend on whether the first or the second argument of the pairing $\mathrm{e}(P, Q)$ is the secret. The reason is that P parameterizes functions such as $\mathrm{mil}_{r,P}(x, y)$ or $T_P(x, y)$ that occur during the pairing computation while these functions are evaluated at Q. Furthermore, we see from Definition 3.8, that these functions are not symmetric in P and Q. In order to give a unified description for either case, we sometimes move P from a parameter to an argument, or more generally:

Definition 5.7. *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be algebraic sets defined over $\mathbb{F}_q$. Furthermore, let $\{f_P : \mathcal{D}_2 \to \mathbb{F}_q\}_{P \in \mathcal{D}_1}$ be a family of maps. We identify this family with the map*

$$\begin{aligned} f : \mathcal{D}_1 \times \mathcal{D}_2 &\to \mathbb{F}_q \\ (P_1, P_2) &\mapsto f_{P_1}(P_2). \end{aligned}$$

Furthermore, for $\mathcal{D}_1 \subseteq \mathbb{A}^n(\mathbb{F}_q)$, $\mathcal{D}_2 \subseteq \mathbb{A}^m(\mathbb{F}_q)$, $P_1 = (x_{1,1}, \ldots, x_{1,n})$, and $P_2 = (x_{2,1}, \ldots, x_{2,m})$ we also write

$$f(x_{1,1}, \ldots, x_{1,n}; x_{2,1}, \ldots x_{2,m}) = f(P_1, P_2).$$

5.2.2. Steps of the analysis

After we defined an initial model $\mathcal{F}$ for a secret S according to Definition 5.6 we perform a sequence of model transformations. The goal is to end up with a model $\mathcal{F}'$ of low degree that contains only polynomials. Hence, this new model defines an algebraic set $V(\mathcal{F}')$ that contains S, or more precisely, that contains an element $(\alpha_1, \ldots, \alpha_n) \in V(\mathcal{F}')$ such that S can be efficiently computed from this element. If the model contains enough independent relations such that $V(\mathcal{F}')$ is of dimension zero and furthermore, if the degree of the model $\mathcal{F}'$ is small enough, we can use standard tools to enumerate all elements of $V(\mathcal{F}')$ and eventually compute S.

We now give an overview of the single steps of the analysis. Later, in Section 5.3, we provide more details on the individual steps.

1. **Oracle queries:** We query the oracle from Definition 5.5 for our fault attack to collect a set of erroneous pairing results in $\mathbb{F}_{q^k}$.

   For details, we refer to Section 5.3.1.

2. **Initial model:** Based on the erroneous pairing results from the previous step, we define an initial model $\mathcal{F} \subset \mathbb{F}_{q^k}(x_1, \ldots, x_n)$ for the secret S (see Definition 5.6).

   For details, we refer to Section 5.3.2.

3. **Model transformations:** To compute S from $\mathcal{F}$, we first perform several transformations on the elements of $\mathcal{F}$ as outlined in the next steps.

   a) **Combine exponents:** Let $\mathcal{F} \subseteq \mathbb{F}_{q^k}(x_1, \ldots, x_n)$ with elements of the form
   $$f(x_1, \ldots, x_n)^{e_1} - a_1$$
   $$\ldots$$
   $$f(x_1, \ldots, x_n)^{e_m} - a_m.$$
   In this step, we combine these functions to one function $f(x_1, \ldots, x_n)^e - a$ with $e = \gcd\left(q^k - 1, e_1, \ldots, e_m\right)$.

   For details, we refer to Section 5.3.3.

   b) **Rescale exponents:** Let $\mathcal{F} \subseteq \mathbb{F}_{q^k}(x_1, \ldots, x_n)$ with $f(x_1, \ldots, x_n)^e - a \in \mathcal{F}$. Assume there exists a $u \in \mathbb{Z}$ with $u \neq \mathrm{ord}(a)$ and $\mathrm{w}_q(ue) \ll \mathrm{w}_q(e)$ (cf. Definition 2.1). Then we replace $f(x_1, \ldots, x_n)^e - a$ with the function $f(x_1, \ldots, x_n)^{ue} - a^u$.

   For details, we refer to Section 5.3.4.

   c) **Split exponents:** Let $\mathcal{F} \subseteq \mathbb{F}_{q^k}(x_1, \ldots, x_n)$ with $f(x_1, \ldots, x_n)^e - a \in \mathcal{F}$. In this step, we first factor e into $e = lh$ with small $\mathrm{w}_q(l)$ and small h. Then we compute $b_1, \ldots, b_h \in \mathbb{F}_{q^k}$ with $b_i^h = a$. Finally, we continue with h copies $\mathcal{F}_1, \ldots, \mathcal{F}_h$ of $\mathcal{F}$. In the copy $\mathcal{F}_i$ we replace $f(x_1, \ldots, x_n)^e - a$ with $f(x_1, \ldots, x_n)^l - b_i$. We perform the subsequent steps for each copy.

   For details, we refer to Section 5.3.5.

   d) **Transformation to the base field:** Let $\mathcal{F} \subseteq \mathbb{F}_{q^k}(x_1, \ldots, x_n)$. In this step, we clear denominators and use Weil restriction of scalars to transform $\mathcal{F}$ into a model in $\mathbb{F}_q[x_{1,1}, \ldots, x_{n,k}]$ with kn variables.

   For details, we refer to Section 5.3.6.

   e) **Fix variables:** Let $\mathcal{F} \subseteq \mathbb{F}_q[x_{1,1}, \ldots, x_{n,k}]$. First, we fix variables from $\{x_{1,1}, \ldots, x_{n,k}\}$ that represent elements that are either known to us or can be guessed with high probability. Then, we proceed with one copy of $\mathcal{F}$ for each guess.

   For details, we refer to Section 5.3.7.

4. **Compute solutions:** Let $\mathcal{F}_1, \mathcal{F}_2, \ldots \subseteq \mathbb{F}_q[x_{1,1}, \ldots, x_{n,k}]$ be the outcome of our transformations. In this step, we first compute
   $$X = \bigcup_{i \geq 1} V(\mathcal{F}_i)(\mathbb{F}_q).$$

Then we compute candidates for the secret S from X.

For details, we refer to Section 5.3.8.

5. **Verify candidates:** In this step, we verify each candidate of the previous step to identify S. For the verification, we can use the corresponding public key or a non-erroneous pairing computation.

   For details, we refer to Section 5.3.9.

5.3. Background information on each step of the framework

In this section, we give details on the individual steps of the analysis that we introduced in the previous section. An important measure that determines the efficiency of step 4 in Section 5.2.2 are the number of monomials that occur in $\mathcal{F}_1, \mathcal{F}_2, \ldots$. We can bound the number of monomials based on the degree and the number of variables that occur in $\mathcal{F}_1, \mathcal{F}_2, \ldots$:

Lemma 5.8. *Let $f(x_1, \ldots, x_n)$ be a polynomial of degree $\deg(f) = d$. Then*

$$\#\operatorname{mon}(f) \leq \binom{d+n}{d}.$$

Proof. Consider the set $\mathcal{S} = \{1, x_1, \ldots, x_n\}$ of size $n+1$. We can think of a monomial as a multiset $\mathcal{M}$ of size d over the elements in $\mathcal{S}$. Multiset, because every variable can occur multiple times in a monomial and because the order of the variables does not matter. Furthermore, the 1 elements in $\mathcal{M}$ allow monomials of degree less than d. The number of multisets of size d over a set of size $n+1$ is given as

$$\binom{d+(n+1)-1}{d}.$$

□

The common rationale behind the individual steps of our analysis therefore is to reduce the degree of the models while keeping the number of variables as small as possible.

5.3.1. Oracle queries

In step 1 of Section 5.2.2, we query our fault oracle, i.e., we perform our fault attack to gather information about the secret S. The more queries we issue on independent inputs, the more information we potentially obtain on S. Furthermore, more constraints on S will potentially increase the efficiency of step 4.

5.3.2. Initial model

In step 2 of Section 5.2.2, we define an initial model $\mathcal{F}$ for a secret S according to Definition 5.6. To end up with a model of small degree in step 4, we already need to define the model in this step such that the number of its variables and its degree are as small as possible. How to define the model strongly depends on the faults we introduce in the attack, but we provide examples that cover all attacks we are aware of.

In our examples, we always consider the reduced Tate pairing from Definition 2.33. For groups $\mathcal{G}_1 \subseteq (\mathbb{F}_q)[r]$ and $\mathcal{G}_2 \subseteq E(\mathbb{F}_{q^k})[r]$ the pairing can be computed as $\mathrm{e}(P,Q) = \mathrm{mil}_{r,P}(Q)^{(q^k-1)/r}$. We first concentrate on the Miller step $\mathrm{mil}_{r,P}(Q)$ of the pairing computation and handle the final exponentiation with $\left(q^k-1\right)/r$ at the end of this section.

Model for attacks without faults in $\mathbb{F}_q$

In this section, we consider fault attacks where the initial model $\mathcal{F}$ consists of functions of the form

$$f(x_1, y_1; x_2, y_2)^c - b \text{ with } b = b_1^{\xi_1} b_2^{\xi_2} \tag{5.1}$$

for $b_1, b_2 \in \mathbb{F}_{q^k}$ and $c, \xi_1, \xi_1 \in \mathbb{Z}$.

Example 5.9. Possibly the simplest example is an attack where the loop of Algorithm 3.4 is terminated in the first iteration after Line 3 has been computed and where the final exponentiation is completely skipped. This attack implements an oracle for the function $f(x_1, y_1; x_2, y_2) = \mathrm{mil}_{2,P_1}(P_2)$ with $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$.

For the result $b_1 = \mathrm{mil}_{2,P}(Q)$ of an oracle query on P and Q we follow the canonical approach described in Section 5.2.1 to define the model by

$$f(x_1, y_1; x_2, y_2) - b_1 \tag{5.2}$$

and the Weierstrass equation (2.3) of the curve. With $\xi_1 = 1$, $\xi_2 = 0$, $c = 1$ the function (5.2) is of the form (5.1).

For $S \in \{P, Q\}$ this results in a model for S, because for the query on P and Q, (P, Q) is a root of (5.2) and obviously we can efficiently compute S from this tuple.

We presented an attack with an initial model similar to this example in [Blö+14]. For a detailed description, see also Section 6.2. Now, we give a more complicated example where $c \neq 1$ in (5.1):

Example 5.10. We assume an attack with two erroneous pairing executions. In both executions, the final exponentiation is completely skipped. For the first execution, we assume that no fault is introduced during the computation of Algorithm 3.4. For the second execution, we assume that a fault is introduced such that

the multiplication in Line 3 with $T_R(Q)/V_{2R}(Q)$ in iteration i of Algorithm 3.4 is skipped. Let R_i be the value of R at the beginning of iteration i in Algorithm 3.4, i.e., $R_i = t_i P$ for some $t_i \in \mathbb{Z}$. With $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ this attack implements an oracle for the functions

$$f_1(x_1, y_1; x_2, y_2) = \mathrm{mil}_{r,P_1}(P_2)$$
$$f_2(x_1, y_1; x_2, y_2) = \mathrm{mil}_{r,P_1}(P_2) \left(\frac{V_{2t_iP_1}(P_2)}{T_{t_iP_1}(P_2)} \right)^{2^{N-i-1}}.$$

To see this, note that the factor $T_{R_i}(Q)/V_{2R_i}(Q)$ is squared for $N - i - 1$ times for the computation of $\mathrm{mil}_{r,P}(Q)$ and that this factor is removed for the second execution.

Let $b_1 = f_1(P, Q)$ and $b_2 = f_2(P, Q)$ be the result of oracle queries on input P and Q. We introduce variables x_3, y_3 that represent R_i and with b_1 and b_2 we define the model $\mathcal{F}$ by the function

$$\left(\frac{T(x_3, y_3; x_2, y_2)}{V([2](x_3, y_3); x_2, y_2)} \right)^{2^{N-i-1}} - \frac{b_1}{b_2} \tag{5.3}$$

together with the Weierstrass equation (2.3). Remember that $[2] : E \to E$ denotes the multiplication-by-2 map and that this map is given by rational functions (cf. Definition 2.9). Finally, with $c = 2^{N-i-1}$, $\xi_1 = 1$, and $\xi_2 = -1$ (5.3) is of the form (5.1).

From the definition of f_1, f_2, and R_i, it follows that (R_i, Q) is a root of (5.3). Furthermore P can be efficiently computed on E from R_i with $P = \left(t_i^{-1} \mod r\right) R_i$. Hence we obtain a model in either case $S = Q$ or $S = P$.

Now, we give an example with $\xi_2 = -2$:

Example 5.11. Again, we assume an attack with two erroneous pairing executions. For both executions, the final exponentiation is skipped. For the first execution, we assume that no fault is introduced during the computation of Algorithm 3.4. For the second execution, we assume that a fault is introduced such that the last iteration of Algorithm 3.4 is completely skipped.

With $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, this attack implements an oracle for the functions

$$f_1(x_1, y_1; x_2, y_2) = \mathrm{mil}_{r,P_1}(P_2)$$
$$f_2(x_1, y_1; x_2, y_2) = \mathrm{mil}_{(r-1)/2,P_1}(P_2).$$

To see this, note that Algorithm 3.4 processes r from the MSB to the LSB. Hence, a skip of the last iteration corresponds to skipping the LSB of r. Because r is an odd prime it holds that the erroneous computation of Algorithm 3.4 evaluates $\mathrm{mil}_{(r-1)/2,P_1}(P_2)$.

Let $b_1 = f_1(P, Q)$ and $b_2 = f_2(P, Q)$ be the result of oracle queries on input P and Q. For $R_{N-1} = ((r-1)/2)P$ we introduce variables x_3, y_3 and with b_1 and b_2 we define the model $\mathcal{F}$ by the function

$$T(x_3, y_3; x_2, y_2) - b_1 b_2^{-2} \tag{5.4}$$

and (2.3). With $c = 1$, $\xi_1 = 1$, and $\xi_2 = -2$ the function ion (5.4) is of the form (5.1).

We claim that (R_{N-1}, Q) is a zero of (5.4). To see this, note that R_{N-1} is the value of R at the beginning of iteration $N-1$ in Algorithm 3.4. Because P is of odd order r it holds that $2R_{N-1} = -P$. By comparing divisors, we obtain $V_{2R_{N-1}} = L_{P,2R_{N-1}}$ and $V_{2R_{N-1}+P} = 1$. Hence, it holds that

$$b_1 = b_2^2 T_{R_{N-1}}(Q)$$

and it follows that $b_1 b_2^{-2} = T_{R_{N-1}}(Q) = T(R_{N-1}, Q)$.

An attack with an initial model like in this example is given in [Mra09].

Model for attacks with faults in $\mathbb{F}_q$

We also consider faults that modify finite field elements in $\mathbb{F}_q$ according to a random distribution. If we are not able to predict the value of the modified field elements, we need to represent them by additional variables in the model. Therefore, we consider models $\mathcal{F} \subseteq \mathbb{F}_{q^k}(x_1, y_1, x_2, y_2, x_5, \ldots, x_n)$ where $x_5, \ldots, x_n$ represent the faults in $\mathbb{F}_q$. For most of the practical attacks, only one fault in $\mathbb{F}_q$ and hence, only one additional variable x_5 is required. For an example, we modify Example 5.10:

Example 5.12. We assume an attack with two erroneous pairing executions. For both executions, the final exponentiation is completely skipped. For the first execution, we assume that no fault is introduced during the computation of Algorithm 3.4. For the second execution, we assume that a transient fault is introduced into the x-coordinate of Q in the computation of $T_R(Q)$ in iteration i of Algorithm 3.4, such that $T_R(Q)$ is replaced by $T_R(x(Q) + \delta, y(Q))$ with $\delta \in \mathbb{F}_q$.

Let R_i be the value of R at the beginning of iteration i in Algorithm 3.4, i.e., $R_i = t_i P$ for some $t_i \in \mathbb{Z}$. With $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, this attack implements an oracle for the functions

$$f_1(x_1, y_1; x_2, y_2) = \mathrm{mil}_{r,P_1}(P_2)$$

$$f_2(x_1, y_1; x_2, y_2; x_5) = \mathrm{mil}_{r,P_1}(P_2) \left(\frac{T_{t_1 P_1}(x_2 + x_5, y_2)}{T_{t_i P_1}(x_2, y_2)} \right)^{2^{N-i-1}}.$$

Let $b_1 = f_1(P, Q)$ and $b_2 = f_2(P, Q, \delta)$ be the result of oracle queries on input P, Q and δ. We introduce variables x_3, y_3 that represent R_i and with b_1 and b_2 we define the model $\mathcal{F}$ by

$$\left(\frac{T(x_3, y_3; x_2, y_2)}{T(x_3, y_3; x_2 + x_5, y_2)} \right)^{2^{N-i-1}} - \frac{b_1}{b_2} \tag{5.5}$$

and (2.3). With $c = 2^{N-i-1}$, $\xi_1 = 1$, and $\xi_2 = -1$ the function (5.5) is of the form (5.1) but with an additional variable x_5.

From the definition of f_1, f_2, and R_i, it follows that (R_i, Q, δ) is a root of (5.3). Furthermore P can be efficiently computed from R_i on E by $P = \left(t_i^{-1} \mod r\right) R_i$. Hence we obtain a model in either case $S = Q$ or $S = P$.

An attack with an initial model like in this example is given in [WS07] . Another case where additional variables are introduced is an attack on masked implementations. Here, the mask as an element defined over $\mathbb{F}_{q^k}$, is represented by variables. Examples are given in [PSM11; MF15].

Extend models to the final exponentiation

So far, we always assumed that the complete final exponentiation is skipped in the attacks and hence, we did not consider it in the definition of the model. In this case, our initial model was always given by a function of the form (5.1).

It is straight forward to extend our models to the case where a fault during the final exponentiation modifies the exponent into an erroneous exponent $\tilde{e}$. We simply need to replace a function of the form $f(x_1, \ldots, x_n)^c - b \in \mathcal{F}$ from (5.1) with $f(x_1, \ldots, x_n)^{c\tilde{e}} - b^{\tilde{e}}$. This transforms a model for an attack where the complete final exponentiation is skipped into a model where the exponent is modified into $\tilde{e}$. For the additional case where a fault is introduced in $\mathbb{F}_q$ during the exponentiation we give an example in Section 5.4.2 .

5.3.3. Combine exponents

In step 3.a of Section 5.2.2, we invert coprime factors of the occurring exponents to reduce the degree of the model. Let $\mathcal{F} \subseteq \mathbb{F}_{q^k}(x_1, \ldots, x_n)$ be a model for the secret S. Suppose, $\mathcal{F}$ contains a set of functions of the form

$$\begin{aligned} f(x_1, \ldots, x_n)^{e_1} - a_1 \\ f(x_1, \ldots, x_n)^{e_2} - a_2 \\ \ldots \\ f(x_1, \ldots, x_n)^{e_m} - a_m \end{aligned} \tag{5.6}$$

with $a_i = b^{e_i}$ for a fixed $b \in \mathbb{F}_{q^k}$.

In this step, we replace the functions in (5.6) by a single function $f(x_1, \ldots, x_n)^e - a$ where $e = \gcd\left(q^k - 1, e_1, \ldots, e_m\right)$. We can determine a based on the extended Euclidean algorithm (EEA):

Lemma 5.13. *Let $a_1, \ldots, a_m \in \mathbb{F}_{q^k}^*$ and $e_1, \ldots, e_m \in \mathbb{N}$. Define $e = \gcd(q^k - 1, e_1, \ldots, e_m)$. Then, there are $u_0, u_1, \ldots, u_m \in \mathbb{Z}$ with $e = u_0(q^k - 1) + u_1 e_1 + \ldots + u_m e_m$. Furthermore, the system*

$$\begin{aligned} x^{e_1} &= a_1 \\ x^{e_2} &= a_2 \\ &\ldots \\ x^{e_m} &= a_m \end{aligned} \tag{5.7}$$

has either no or exactly e solutions in $\mathbb{F}_{q^k}$. In the latter case, define $a = \prod_{i=1}^m a_i^{u_i}$. Then the solutions of (5.7) are the e-th roots of a in $\mathbb{F}_{q^k}$.

Proof. By repeated application of the EEA we can compute $u_0, \ldots, u_m$ as required. Now assume (5.7) has a solution $x_0 \in \mathbb{F}_{q^k}$. Let $x_0' = x_0\xi$ be any element in $\mathbb{F}_{q^k}^*$. It holds that $(x_0\xi)^{e_i} = a_i\xi^{e_i}$. Furthermore $a_i\xi^{e_i} = a_i$ for all $i \in [1, m]$ if and only if $\mathrm{ord}(\xi)|e$. Hence $\xi \in \mu_e$, the e-th roots of unity. Because $e|q^k - 1$ it holds that $\mu_e \subseteq \mathbb{F}_{q^k}^*$. Hence, if (5.7) has one solution x_0, then all solutions are given by the coset $x_0\mu_e$.

Finally,

$$x_0^e = x_0^{u_0(q^k-1)} \prod_{i=1}^{m} x_0^{u_i e_i} = \prod_{i=1}^{m} a_i^{u_i} = a.$$

□

This lemma shows that a simultaneous zero of (5.6) is also a zero of the function $f(x_1, \ldots, x_n)^e - a$. Hence, replacing (5.6) with $f(x_1, \ldots, x_n)^e - a$ will transform a model $\mathcal{F}$ for a secret S into a new model for S.

The typical application where our model contains a system like (5.6) occurs in an attack on the final exponent. Suppose b is the erroneous result of the Miller algorithm on fixed inputs P and Q, modeled by $f(x_1, \ldots, x_n)$, and $e_1, \ldots, e_m$ are exponents resulting from different faults during the final exponentiation. Hence, we obtain $a_1, \ldots, a_m$ with $a_i = b^{e_i}$ as results of the erroneous pairing computations.

5.3.4. Rescale exponents

In step 3.b of Section 5.2.2, we reduce the weight (see Definition 2.1) of exponents because their weight will determine the degree of the model in step 3.d. We reduce the weight of an exponent by multiplication with an appropriate factor u.

Let $\mathcal{F} \subseteq \mathbb{F}_{q^k}(x_1, \ldots, x_n)$. Consider a function in $\mathcal{F}$ of the form

$$f(x_1, \ldots, x_n)^e - a \tag{5.8}$$

with $a \in \mathbb{F}_{q^k}$. First we compute $u \in \mathbb{Z}$ such that $\mathrm{w}_q(ue) \ll \mathrm{w}_q(e)$ and $ue \neq q^k - 1$. Then we replace (5.8) in $\mathcal{F}$ with the function

$$f(x_1, \ldots, x_n)^{ue} - a^u.$$

If $\mathcal{F}$ is a model for the secret S, then the result of this transformation obviously is also a model for S. If $\gcd(q^k - 1, u) \neq 1$, the new model might have additional solutions. But as long as we obtain a non-trivial relation, i.e., $ue \neq q^k - 1$, we can counteract this by adding more independent equations to the model based on additional queries to the oracle in step 1 of Section 5.2.2.

In most applications, we can find u as a divisor of $(q^k - 1)/e$. But [Ver08] also give a more general approach to find u based on lattices. They define the lattice

that is spanned by the rows of the matrix

$$B = \begin{pmatrix} e & 0 & 0 & \dots & 0 \\ -q & 1 & 0 & \dots & 0 \\ -q^2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -q^{k-1} & 0 & 0 & \dots & 1 \end{pmatrix}.$$

The next lemma shows that we can find multiples of e with small weight based on this lattice:

Lemma 5.14. *Let v be a vector in the lattice spanned by the rows of B. For $w = (1, q, q^2, \dots, q^{k-1})$ with transpose w^T it holds that $e | vw^T$.*

Proof. Let $v = (v_0, \dots, v_{k-1})$. From the form of B it follows that $v_0 = ne - \sum_{i=1}^{k-1} v_i q^i$ for some $n \in \mathbb{Z}$. Hence, $ne = \sum_{i=0}^{k-1} v_i q^i = vw^T$ and therefore $e | vw^T$. □

By definition of w, the vector v defines a q-ary representation of vw^T. According to Definition 2.1, a short v with small ℓ_1-norm provides a light multiple of e.

5.3.5. Split exponents

In step 3.c of Section 5.2.2 that is complementary to step 3.b, we invert a small factor of the exponents to decrease the degree of the resulting models.

Let $\mathcal{F} \subseteq \mathbb{F}_{q^k}(x_1, \dots, x_n)$ and assume $\mathcal{F}$ contains a function of the form

$$f(x_1, \dots, x_n)^{lh} - a \tag{5.9}$$

with light l (cf. Definition 2.1) and small h. Then we compute the h-th roots $b_1, \dots, b_h$ of a. Finally, we replace $\mathcal{F}$ with h copies $\mathcal{F}_1, \dots, \mathcal{F}_h$ where in $\mathcal{F}_i$, we replace (5.9) with

$$f(x_1, \dots, x_n)^{l} - b_i. \tag{5.10}$$

We obtain a new model for the secret S:

Lemma 5.15. *If $\mathcal{F}$ is a model for S, then there exists an $i \in [1, h]$ such that $\mathcal{F}_i$ is a model for S.*

Proof. If $\mathcal{F}$ is a model for S, then there exists a zero $\alpha = (\alpha_1, \dots, \alpha_n)$ of (5.9) such that S can be efficiently computed from α. Let $b = f(\alpha_1, \dots, \alpha_n)^l$. Then $b^h = a$ and hence $b \in \{b_1, \dots, b_h\}$. Therefore, there exists an $i \in [1, h]$ such that α is also a zero of (5.10). Hence, $\mathcal{F}_i$ is a model for S. □

How to choose l and h depends on the concrete application. On the one hand, a large h may help to decrease $\mathrm{w}_q(l)$. This enables us to reduce the degree of the models in step 3.d. On the other hand, a large h introduces more models. This increases the complexity for computing all candidate solutions in step 4. Hence, in this step, we need to find a trade-off between the number of new instances and the reduction of the degree.

5.3.6. Transformation to the base field

In step 3.d of Section 5.2.2, we finally exploit light exponents. According to Lemma 5.8, the number of monomials of a function of the form $f(x_1, \ldots, x_n)^e$ is bounded by

$$\binom{e \deg(f) + n}{e \deg(f)}. \tag{5.11}$$

In this step, we transform a function of the form $f(x_1, \ldots, x_n)^e - a \in \mathbb{F}_{q^k}(x_1, \ldots, x_n)$ into a system of polynomial equations in $\mathbb{F}_q[x_{1,1}, \ldots, x_{1,k}, \ldots, x_{n,1}, \ldots, x_{n,k}]$ that has kn variables. The following theorem summarizes the central property of this transformation. It shows that the degree of the resulting system depends on $\mathrm{w}_q(e)$ rather than on e itself:

Theorem 5.16. *Let $a \in \mathbb{F}_{q^k}$ and*

$$f(x_1, \ldots, x_n)^e - a \in \mathbb{F}_{q^k}(x_1, \ldots, x_n). \tag{5.12}$$

There is a set of polynomials $\mathcal{F} = \{f_1, \ldots, f_k\} \subseteq \mathbb{F}_q[x_{1,1}, \ldots, x_{1,k}, \ldots, x_{n,1}, \ldots, x_{n,k}]$ with degree at most $\mathrm{w}_q(e) \deg(f)$ such that the $\mathbb{F}_{q^k}$-rational zeros of (5.12) are in one-to-one correspondence with the algebraic set $V(\mathcal{F})(\mathbb{F}_q)$.

According to Lemma 5.8, the number of monomials of $\mathcal{F}$ is bounded by

$$\binom{\mathrm{w}_q(e) \deg(f) + kn}{\mathrm{w}_q(e) \deg(f)}.$$

Hence, the theorem shows that we can replace the dependency on e in (5.11) by a dependency on $\mathrm{w}_q(e)$ at the expense of increasing the number of variables from n to kn.

We now perform the transformation that we apply to an element of the form (5.12) in order to give a constructive proof of Theorem 5.16.

Clear denominator

First let $f = g/h$ with $g, h \in \mathbb{F}_{q^k}[x_1, \ldots, x_n]$. Then, we clear denominators of (5.12) and obtain

$$g(x_1, \ldots, x_n)^e - a h(x_1, \ldots, x_n)^e. \tag{5.13}$$

Clear exponent

Now, let $(\epsilon_0, \epsilon_1, \ldots)$ be the q-ary representation of e from Definition 2.1, i.e., in particular $e = \sum_{i \geq 0} \epsilon_i q^i$. We expand (5.13) into

$$\prod_{\epsilon_i > 0} g(x_1, \ldots, x_n)^{\epsilon_i q^i} \prod_{\epsilon_i < 0} h(x_1, \ldots, x_n)^{|\epsilon_i| q^i} - a \prod_{\epsilon_i > 0} h(x_1, \ldots, x_n)^{\epsilon_i q^i} \prod_{\epsilon_i < 0} g(x_1, \ldots, x_n)^{|\epsilon_i| q^i}.$$

Note that exponentiation with q^i is a linear map on $\mathbb{F}_{q^k}$. Based on this, we apply the exponents to the coefficients of g and h. With the q^i-th power Frobenius maps $\pi_{q^i} : \mathbb{A}^n \to \mathbb{A}^n$ we obtain:

$$\prod_{\epsilon_i>0} g^{q^i}(\pi_{q^i}(x_1,\ldots,x_n))^{\epsilon_i} \prod_{\epsilon_i<0} h^{q^i}(\pi_{q^i}(x_1,\ldots,x_n))^{|\epsilon_i|} - a \prod_{\epsilon_i>0} h^{q^i}(\pi_{q^i}(x_1,\ldots,x_n))^{\epsilon_i} \prod_{\epsilon_i<0} g^{q^i}(\pi_{q^i}(x_1,\ldots,x_n))^{|\epsilon_i|}. \quad (5.14)$$

Weil restriction of scalars

Now we construct the Weil restriction of scalars of the algebraic set defined by (5.14) with respect to $\mathbb{F}_{q^k}/\mathbb{F}_q$. Let $\{\theta_1,\ldots,\theta_k\}$ be a basis for $\mathbb{F}_{q^k}/\mathbb{F}_q$ and let ϕ be as in Definition 2.2. We abbreviate $\underline{x} = x_{1,1},\ldots,x_{1,k},\ldots,x_{n,1},\ldots,x_{n,k}$. According to Definition 2.2 we replace π_{q^i} with $\pi_{q^i} \circ \phi = \phi^{q^i} \circ \pi_{q^i}$ to obtain

$$\prod_{\epsilon_i>0} g^{q^i}(\phi^{q^i}(\pi_{q^i}(\underline{x})))^{\epsilon_i} \prod_{\epsilon_i<0} h^{q^i}(\phi^{q^i}(\pi_{q^i}(\underline{x})))^{|\epsilon_i|} - a \prod_{\epsilon_i>0} h^{q^i}(\phi^{q^i}(\pi_{q^i}(\underline{x})))^{\epsilon_i} \prod_{\epsilon_i<0} g^{q^i}(\phi^{q^i}(\pi_{q^i}(\underline{x})))^{|\epsilon_i|} \quad (5.15)$$

in $\mathbb{F}_{q^k}[x_{1,1},\ldots,x_{1,k},\ldots,x_{n,1},\ldots,x_{n,k}]$. Note that ϕ is a bijection from $\mathbb{A}^{kn}(\mathbb{F}_q)$ to $\mathbb{A}^n(\mathbb{F}_{q^k})$ and hence, the $\mathbb{F}_{q^k}$-rational roots of (5.14) are in one-to-one correspondence with the $\mathbb{F}_q$-rational roots of (5.15).

Because $\pi_{q^i}(P) = P$ for every $P \in \mathbb{A}^{nk}(\mathbb{F}_q)$ we keep the $\mathbb{F}_q$-rational roots fixed when we drop the π_{q^i} and replace (5.15) with

$$\prod_{\epsilon_i>0} g^{q^i}(\phi^{q^i}(\underline{x}))^{\epsilon_i} \prod_{\epsilon_i<0} h^{q^i}(\phi^{q^i}(\underline{x}))^{|\epsilon_i|} - a \prod_{\epsilon_i>0} h^{q^i}(\phi^{q^i}(\underline{x}))^{\epsilon_i} \prod_{\epsilon_i<0} g^{q^i}(\phi^{q^i}(\underline{x}))^{|\epsilon_i|}. \quad (5.16)$$

Like in Definition 2.2, we sort terms of (5.16) with respect to the basis $\{\theta_1,\ldots,\theta_k\}$ to obtain

$$\sum_{j=1}^{k} f_j(\underline{x})\theta_j \text{ with } f_j(\underline{x}) \in \mathbb{F}_q[\underline{x}]. \quad (5.17)$$

With $\mathcal{F} = \{f_1,\ldots,f_k\}$ the algebraic set $Y = V(\mathcal{F}) \subseteq \mathbb{A}^{kn}$ is the Weil restriction of scalars of the algebraic set X that is defined by (5.16). By Theorem 2.3, the map ϕ defines a bijection from $Y(\mathbb{F}_q)$ to $X(\mathbb{F}_{q^k})$ that provides us with the one-to-one correspondence of Theorem 5.16.

Finally, from (5.16) we see that

$$\max\{\deg(f_1),\ldots,\deg(f_k)\} \leq \max\left\{\deg(g)\sum_{\epsilon_i>0}\epsilon_i + \deg(h)\sum_{\epsilon_i<0}|\epsilon_i|\,,\ \deg(h)\sum_{\epsilon_i>0}\epsilon_i + \deg(g)\sum_{\epsilon_i<0}|\epsilon_i|\right\}.$$

By definition, $\deg(g) \leq \deg(f)$ and $\deg(h) \leq \deg(f)$. Hence, with Definition 2.1 we obtain

$$\max\{\deg(f_1), \ldots, \deg(f_k)\} \leq \deg(f) \sum_{i \geq 0} |\epsilon_i| = \deg(f)\, \mathrm{w}_q(e).$$

This concludes the proof of Theorem 5.16.

Remark 5.17. Let $i \in [1, n]$ and assume the variable x_i in (5.12) represents an element a of a proper subfield $\mathbb{F}_{q^d} \subset \mathbb{F}_{q^k}$. Without loss of generality, let $\{\theta_1, \ldots, \theta_d\}$ be a basis for $\mathbb{F}_{q^d}/\mathbb{F}_q$. Then, $a \in \mathbb{F}_{q^d}$ is represented as $a = \sum_{i=1}^{d} a_i \theta_i$ with $a_i \in \mathbb{F}_q$. Hence, we can restrict to the $\mathbb{F}_{q^d}$-rational solutions for x_i by setting $x_{i,d+1}, \ldots, x_{i,k}$ to 0 in $\mathcal{F} = \{f_1, \ldots, f_k\}$. This reduces the number of variables and hence the number of monomials.

The case where S is defined over a proper subfield $\mathbb{F}_{q^d}$ of $\mathbb{F}_{q^k}$ often occurs in practice. Take the reduced Tate pairing for an example. Here, the first argument P of the pairing is already defined over $\mathbb{F}_q$: i.e., if $S = P$ we even get $d = 1$ for the variables that represent P. Furthermore for Type 3 pairings (cf. Definition 3.1) and with Theorem 2.21, the second argument Q can be represented based on a twist E' of E. Then, for $S = Q$ it holds that $Q \in E'(\mathbb{F}_{q^d})$ if the twist is of degree k/d. For example, in the case of BN curves with $k = 12$ that admit a twist of degree 6, it holds that $Q \in E'(\mathbb{F}_{q^2})$. Note that we use these properties in our attacks from Chapter 6. Finally, for variables that represent faults in $\mathbb{F}_q$ we also obtain $d = 1$.

5.3.7. Fix variables

Let $\mathcal{F} \subseteq \mathbb{F}_q[x_{1,1}, \ldots, x_{n,k}]$. In step 3.e of Section 5.2.2, we fix variables of $\mathcal{F}$ that we either know, or that we can guess with high probability. Then we proceed with one copy of $\mathcal{F}$ for each guess. There are two typical applications where we can fix some of the variables:

To define our initial model in Section 5.3.2 independently from the situation whether the argument P or the argument Q of the pairing is secret, we introduced variables for both of them. Now, before computing solutions for the secret in step 4, we fix all variables that depend on the public argument of the pairing.

Furthermore, in an attack with faults in $\mathbb{F}_q$, we introduced variables that represent these faults in the initial model. If there are enough independent equations in the model $\mathcal{F}$ then the corresponding algebraic set $V(\mathcal{F})$ has zero dimension and we can solve the system to obtain candidates for the secret and for the faults in $\mathbb{F}_q$. But in some situations, it is beneficial to remove the variables that represent faults in $\mathbb{F}_q$ by guessing their assignment. Of course, we can only guess assignments if the process that introduces the faults in $\mathbb{F}_q$ is precise and hence has low entropy. Examples are byte faults where one byte of the representation of an element in $\mathbb{F}_q$ is disturbed. Guessing assignments might turn $V(\mathcal{F})$ of positive dimension into an algebraic set with zero dimension. Furthermore, even for a zero dimensional algebraic set, reducing the number of variables can speed up the enumeration of its elements by decreasing the number of monomials.

5.3.8. Compute solutions

Let $\mathcal{F}_1, \mathcal{F}_2, \ldots \subseteq \mathbb{F}_q[x_{1,1}, \ldots, x_{n,k}]$. In step 4 we first compute

$$X = \bigcup_{i \geq 1} V(\mathcal{F}_i)(\mathbb{F}_q).$$

Assume there exists an $i \geq 1$ such that $\mathcal{F}_i$ is a model for S. According to Definition 5.6, X then contains an element $\alpha = (\alpha_{1,1}, \ldots, \alpha_{n,k})$ such that S can be computed from α. Hence, we are able to compute candidates for S from X.

To compute the elements in $V(\mathcal{F}_i)(\mathbb{F}_q)$, we search for the simultaneous $\mathbb{F}_q$-rational roots of elements in $\mathcal{F}_i$. There are various methods to solve systems of multivariate equations and we do not discuss them here. For our applications we can typically use standard methods that are part of a computer algebra system (CAS) like resultants or Groebner bases techniques (see, e.g., [CLO96]).

Note that the variables that represent the y-coordinate of the secret can be eliminated by using the Weierstrass equation (2.3) of the curve. Either this happens implicitly in the process of solving $\mathcal{F}_i$ or, prior to step 3.d of Section 5.2.2, we compute the resultant of every polynomial in the model with $y^2 - x^3 - a_4x - a_6$.

Computing $V(\mathcal{F}_i)(\mathbb{F}_q)$ may fail for mainly two reasons. One reason is that the model $\mathcal{F}_i$ defines an algebraic set of dimension greater than 0 because the defining equations are not algebraically independent. In this case, there are a potentially q or more satisfying assignments. Furthermore, if $\mathcal{F}_i$ contains a large number of monomials, it might be computationally infeasible to compute $V(\mathcal{F}_i)(\mathbb{F}_q)$ even if it is of dimension 0. In both cases, additional constraints from oracle queries on independent public arguments in step 1 of Section 5.2.2 might solve the problem.

5.3.9. Verify candidates

In step 5 of Section 5.2.2, we identify the secret S among the candidates that were computed in step 4. This is possible because we are in a public key setting where the public key corresponding to S can be used to test the functionality of candidate solutions.

5.4. Application of the framework to previous attacks

In this section we demonstrate our framework by analyzing previous attacks from the literature. We consider two attacks:

- In Section 5.4.1 we analyze the attack from [PV06] on the Eta pairing to demonstrate step 2 and step 3.d of the framework from Section 5.2.2.
- In Section 5.4.2 we analyze the attack from [LFG13] on the final exponentiation of the Tate pairing to demonstrate step 2, step 3.b, step 3.d, and step 3.e of the framework from Section 5.2.2.

In Chapter 6, we provide examples that demonstrate also step 3.a, and step 3.c of Section 5.2.2.

5.4.1. Loop bound attack on the Duursma-Lee algorithm

In [PV06] an attack on the Duursma-Lee algorithm [DL03] for computing the Eta pairing in characteristic 2 or 3 is presented. It is the first published fault attack on pairings. Here, this attack serves as an example to demonstrate our framework, especially step 2 and step 3.d from Section 5.2.2. We explain the case of characteristic 3 but the attack for characteristic 2 is similar. We will see that the low weight of the final exponent $e = q^3 - 1$ in base q of this particular pairing enables an efficient first order attack.

Background information on the attacked implementation

We now give the relevant details on the Eta pairing in characteristic 3. For more background we refer to [Bar+07], Section 5 and to [DL03]. Let $E : y^2 = x^3 - x + a_6$ be an elliptic curve defined over $\mathbb{F}_3$. For $\mathbb{F}_q$ with $q = 3^m$ and $\gcd(m, 6) = 1$ let $\epsilon = 1$ if $m \mod 12 \in \{1, 11\}$ and $\epsilon = -1$ if $m \mod 12 \in \{5, 7\}$. It holds that $n = \#E(\mathbb{F}_q) = 3^m + 1 + \epsilon 3^{(m+1)/2}$ and that the embedding degree of q and n is $k = 6$. We define $\mathbb{F}_{q^3} = \mathbb{F}_q(\rho)$, $\mathbb{F}_{q^2} = \mathbb{F}_q(\sigma)$, and $\mathbb{F}_{q^6} = \mathbb{F}_q(\rho, \sigma)$ with $\rho^3 - \rho - a_6 = 0$ and $\sigma^2 + 1 = 0$. With $\mu_i(x_1, x_2) = x_1 + x_2 + a_6$, define

$$g(x_1, y_1; x_2, y_2) = -y_1 y_2 \sigma - \mu_i(x_1, x_2)^2 - \mu_i(x_1, x_2)\rho - \rho^2. \tag{5.18}$$

The Duursma-Lee algorithm computes the Eta pairing $\eta : E(\mathbb{F}_q)[n] \times E(\mathbb{F}_q)[n] \to \mathbb{F}_{q^k}$ in characteristic 3 as a product, followed by an exponentiation with $q^3 - 1$ (cf. Algorithm 1 of [PV06]):

$$\eta(P, Q) = \left(\prod_{i=1}^{m} g\left(x(P)^{3^i}, y(P)^{3^i}; x(Q)^{3^{-i+1}}, y(Q)^{3^{-i+1}} \right) \right)^{q^3-1}. \tag{5.19}$$

Background information on the attack

The attack assumes a fault that modifies the loop bound m in (5.19) into $m + \delta$. From now on, we assume that $\delta \leq m$. Including pairing computations without faults the attack implements an oracle for the following functions:

$$\begin{aligned} f_1(x_1, y_1; x_2, y_2) &= \eta(x_1, y_1; x_2, y_2) \\ f_2(x_1, y_1; x_2, y_2) &= \eta(x_1, y_1; x_2, y_2) \left(\prod_{i=m+1}^{m+\delta} g\left(x_1^{3^i}, y_1^{3^i}; x_2^{3^{-i+1}}, y_2^{3^{-i+1}} \right) \right)^{q^3-1}. \end{aligned} \tag{5.20}$$

The pairing computation is executed twice, one time without fault and one time with fault to obtain $a_i = f_i(P, Q)$ for $i \in \{1, 2\}$.

Analysis of the attack

The following theorem summarizes the efficiency of the attack that shows that the analysis of the attack is most efficient for $\delta = 1$:

Theorem 5.18. *Let $S \in \{P, Q\}$. From a_1 and a_2 we can derive a model for S that contains at most 4^δ monomials.*

Initial model: First we define an initial model with degree depending on δ rather than on m. Therefore let $a = a_2/a_1$ and

$$f(x_1, y_1; x_2, y_2) = \prod_{i=1}^{\delta} g\left(x_1^{3^i}, y_1^{3^i}; x_2^{3^{\delta-i+1}}, y_2^{3^{\delta-i+1}}\right). \tag{5.21}$$

We obtain a model for P and Q:

Lemma 5.19. *The function*

$$f(x_1, y_1; x_2, y_2)^{q^3-1} - a \tag{5.22}$$

defines a model for P and Q.

Proof. We claim that $\left(x(P), y(P), x(Q)^{3^{m-\delta}}, y(Q)^{3^{m-\delta}}\right)$ is a root of (5.22). On the one hand it holds that

$$f\left(x(P), y(P); x(Q)^{3^{m-\delta}}, y(Q)^{3^{m-\delta}}\right) = \prod_{i=1}^{\delta} g\left(x(P)^{3^i}, y(P)^{3^i}; x(Q)^{3^{m-i+1}}, y(Q)^{3^{m-i+1}}\right).$$

On the other hand, it follows from the definition of a_1, a_2, and (5.20) that

$$a = \frac{a_2}{a_1} = \prod_{i=1}^{\delta} g\left(x(P)^{3^{m+i}}, y(P)^{3^{m+i}}; x(Q)^{3^{-m-i+1}}, y(Q)^{3^{-m-i+1}}\right)^{q^3-1}.$$

Because the Frobenius map π_q is the identity on $E(\mathbb{F}_q)$ and because $P, Q \in E(\mathbb{F}_q)$ we can drop every occurrence of 3^m in the exponent. By comparing the two products it follows that

$$f\left(x(P), y(P); x(Q)^{3^{m-\delta}}, y(Q)^{3^{m-\delta}}\right)^{q^3-1} = a.$$

Hence, $\left(x(P), y(P), x(Q)^{3^{m-\delta}}, y(Q)^{3^{m-\delta}}\right)$ is a root of (5.22).

Furthermore, $x(Q)$ and $y(Q)$ can be computed efficiently from $x(Q)^{3^{m-\delta}}$ and $y(Q)^{3^{m-\delta}}$ by exponentiation with 3^δ. From Definition 5.6 it follows that (5.22) is a model for P and Q. □

Combine exponents: In this attack, only one exponent, namely $q^3 - 1$ occurs. Furthermore, this exponent divides $q^6 - 1$, the order of $\mathbb{F}_{q^6}^*$, and we cannot invert factors that are coprime to the order of elements in $\mathbb{F}_{q^6}^*$. Hence, this step does not apply for this attack.

Rescale exponents: We see from Definition 2.1 that the weight of $q^3 - 1$ in base q equals $\mathrm{w}_q(q^3 - 1) = 2$. Because this is the smallest weight possible for exponents that are not powers of q, this step does not apply here.

Split exponents: For the same reason as before, this step does not apply.

Transformation to the base field: Now we transform (5.22) into a polynomial system over $\mathbb{F}_q$ as explained in Section 5.3.6. In our special case, $f(x_1, y_1; x_2, y_2)$ is already a polynomial and hence the denominator h of (5.13) can be set to 1. Because $P, Q \in E(\mathbb{F}_q)$ we do not have to introduce additional variables to apply the Weil restriction of scalars (cf. Remark 5.17). We define $\phi(x_1, y_1; x_2, y_2) = (x_1, y_1; x_2, y_2)$ and obtain

$$f^{q^3}(x_1, y_1; x_2, y_2) - af(x_1, y_1; x_2, y_2) \tag{5.23}$$

for (5.16) on page 73.

Finally, we sort terms in (5.23) according to the basis $\{1, \sigma, \rho, \sigma\rho, \rho^2, \sigma\rho^2\}$ of $\mathbb{F}_{q^k}$ as a $\mathbb{F}_q$ vector space and rewrite (5.23) as

$$\sum_{i=0}^{1}\sum_{j=0}^{2} f_{i,j}(x_1, y_1; x_2, y_2)\sigma^i\rho^j$$

with $f_{i,j}(x_1, y_1; x_2, y_2) \in \mathbb{F}_q[x_1, y_1, x_2, y_2]$. This defines the system

$$\mathcal{F} = \{f_{0,0}(x_1, y_1; x_2, y_2), \ldots, f_{1,2}(x_1, y_1; x_2, y_2)\} \subseteq \mathbb{F}_q[x_1, y_1, x_2, y_2] \tag{5.24}$$

of $k = 6$ polynomials.

Fix variables: Define $\tilde{Q} = \pi_{3^{m-\delta}}(Q)$. If $S = P$ we assign $(x_2, y_2) = \tilde{Q}$ and if $S = Q$ we assign $(x_1, y_1) = P$ in $\mathcal{F}$ to obtain polynomials in $\mathbb{F}_q[x_1, y_1]$ or $\mathbb{F}_q[x_2, y_2]$, respectively. From (5.18), we see that $\mathrm{mon}(g(x_1, y_1; x_2, y_2)) = \{1, y_1y_2, x_1, x_2, x_1x_2, x_1^2, x_2^2\}$. Hence, after fixing variables, the number of monomials is bounded by 4. The function $f(x_1, y_1, x_2, y_2)$ from (5.21) is a product of δ polynomials of the form (5.18). It follows that the number of monomials in $\mathcal{F}$ after fixing variables is bounded by 4^δ. This proves Theorem 5.18.

Compute solutions: In this step, we solve the system (5.24) of polynomial equations. With the equation of the curve we have $k + 1 = 7$ polynomial equations for only two variables and it is not required to repeat the attack for adding additional constraints to the model.

Discussion

In [PV06] Page and Vercauteren split the analysis into the two steps exponentiation inversion and Miller inversion. In the first step, they invert the final exponent to find b with $b^{q^3-1} = a$. In the second step, they compute the secret point from b and the equation (5.18) of $g(x_1, y_1; x_2, y_2)$. Note that (5.18) is sparse in the sense that only the basis elements $\{1, \sigma, \rho, \rho^2\}$ occur. In order to be able to invert the final exponent, this special sparse form and the special form of the final exponent is used in [PV06] to identify the $q^3 - 1$-th root of a that is in the image of $g(x_1, y_1; x_2, y_2)$. To be able to exploit the form of $g(x_1, y_1; x_2, y_2)$, Page and Vercauteren require either $\delta = 1$ or they require two faults with $\delta = t$ and $\delta = t \pm 1$.

In our analysis, we combine the complete inversion into one step by applying the exponent $q^3 - 1$ to $f(x_1, y_1; x_2, y_2)$ and expanding f into a product with $\mathrm{w}_q(q^3 - 1) = 2$ factors. Hence, our analysis shows that it is the low weight of the final exponent $e = q^3 - 1$ and not the special structure of (5.18) that makes this attack efficient. Furthermore, this analysis permits an attack with one fault also in the case where $\delta \neq 1$.

We further remark that in the case where the Eta pairing is computed based on the improvements from Section 5.1 of [Bar+07] with halved loop length, the final exponent is given as $(q^4 + q^3 - q - 1)(q \pm 3\sqrt{q} + 1)$. Then, the analysis of [PV06] is not possible because the exponent has not the required form that allows to identify the correct root of a. Because the exponent has a heavy factor $h = q \pm 3\sqrt{q} + 1$ of weight $\mathrm{w}_q(h) = 3\sqrt{q} + 2$ the number of monomials after step 3.d is in the order of $4^{3\sqrt{q}\delta}$. Hence, our analysis is also not feasible.

5.4.2. Inverting the final exponentiation with faults

In [LFG13] Lashermes, Fournier, and Goubin present an approach to invert the final exponentiation with faults. In this section, we describe the attack based on our framework in order to present a structured analysis and to identify the crucial ideas of the attack.

Background information on the attacked implementation

We make a few reasonable assumptions about the implementation of the final exponentiation. For more background, we refer to Section 3.2.3. We assume a finite field $\mathbb{F}_{p^k}$ with even extension degree $k = 2d$. We define $q = p^d$ and hence $\mathbb{F}_{q^2} = \mathbb{F}_{p^k}$. To ease notation, we assume that $\mathbb{F}_{q^2}/\mathbb{F}_q$ is defined by a binomial $x^2 - v$ with root θ. Then $\theta^2 = v$ and $\theta^q = -\theta$.

For the final exponent $(p^k - 1)/r = (q^2 - 1)/r$ of the reduced pairings from Section 2.3.3 with prime r and $r|q+1$ we assume that the implementation computes $a = b^{(q^2-1)/r}$ in two steps (cf. Section 3.2.3):

1. $t \leftarrow b^{q-1}$
2. $a \leftarrow t^{(q+1)/r}$.

Background information on the attack

In the attack, the computation of the exponentiation on input b is executed for several times. In each execution i a single fault $\delta_i \in \mathbb{F}_q$ is introduced into the computation to obtain the output $a_i \in \mathbb{F}_{q^2}$. Let $\mathcal{D} \subseteq \mathbb{F}_q$ be the domain of faults that we introduce in the attack. For precise faults, we obtain small $\mathcal{D}$. For example, for byte faults, it holds that $\#\mathcal{D} \leq 256$.

We define the oracle (cf. Definition 5.5) for this attack based on the following two functions:

$$f_\nu(x, y) = (x^{q-1} + x^{\nu q} y)^{(q+1)/r} \text{ with } \nu \in \{0, 1\}.$$

Then the faults are introduced at the computation of the final exponentiation such that

$$a_0 = f_0(b, 0) \qquad a_1 = f_0(b, \delta_1) \qquad a_2 = f_1(b, \delta_2)$$

for $\delta_1, \delta_2 \in \mathcal{D} \subseteq \mathbb{F}_q$. Hence, a_0 corresponds to the correct execution of the exponentiation on input b. The output a_1 corresponds to a fault $\delta_1 \in \mathbb{F}_q$ that is introduced after the first step of the exponentiation that modifies $t = b^{q-1}$ into $t + \delta_1 = b^{q-1} + \delta_1$. The output a_2 corresponds to a fault $\delta_2 \in \mathbb{F}_q$ during the computation of b^{-1} in the first step of the exponentiation that modifies $b^{q-1} = b^q b^{-1}$ into $b^q(b^{-1} + \delta_2) = b^{q-1} + b^q \delta_2$.

Analysis of the attack

Now we outline the analysis according to Section 5.2.2. The goal is to recover the input of the final exponentiation b based on a_0, a_1, and a_2.

Initial model: We define a model $\mathcal{F} \subseteq \mathbb{F}_{q^2}(x_0, x_1, y_1, y_2)$ for b according to Definition 5.6. Here x_0 represents b, x_1 represents $t = b^{q-1}$, y_1 represents δ_1, and y_2 represents δ_2. The model is given by the following functions:

$$x_0^{q-1} - x_1 \qquad x_1^{q+1} - 1 \tag{5.25}$$

$$(x_1 + y_1)^{(q+1)/r} - a_1 \qquad (x_1 + x_0 x_1 y_2)^{(q+1)/r} - a_2. \tag{5.26}$$

Lemma 5.20. *The functions in* (5.25) *and* (5.26) *define a model for the input b of the final exponentiation with $(q^2 - 1)/r$.*

Proof. We claim that the assignment $(x_0, x_1, y_1, y_2) = (b, b^{q-1}, \delta_1, \delta_2)$ is a simultaneous root of the polynomials in (5.25) and (5.26). Then the claim follows from Definition 5.6.

For $b \in \mathbb{F}_{q^2}$ it holds that $b^{(q-1)(q+1)} = 1$ and hence, (b, b^{q-1}) is a simultaneous root of (5.25). From the definition of a_1 and a_2 we see that $(b, b^{q-1}, \delta_1, \delta_2)$ is simultaneous root of (5.26). □

Hence, $x_0^{q-1} - x_1$ encodes the relation $t = b^{q-1}$ and $x_1^{q+1} - 1$ encodes that $t \in \mu_{q+1}$. Note that we introduced a redundant variable x_1 for x_0^{q-1}. As we will see later, this increases the efficiency of the analysis.

Combine exponents: Because the exponents $q-1$, $q+1$, and $(q+1)/r$ that occur in (5.25) and (5.26) divide $q^2 - 1$, this step does not apply.

Rescale exponents: Now, we rescale the exponent $(q+1)/r$ of (5.26) that has weight $\mathrm{w}_q((q+1)/r) = (q+1)/r$ with the factor r to obtain the exponent $q+1$ of weight $\mathrm{w}_q(q+1) = 2$. We replace (5.26) in $\mathcal{F}$ and obtain the model

$$x_0^{q-1} - x_1 \qquad x_1^{q+1} - 1 \tag{5.27}$$

$$(x_1 + y_1)^{q+1} - a_1^r \qquad (x_1 + x_0 x_1 y_2)^{q+1} - a_2^r. \tag{5.28}$$

The new system has potentially more solutions than the original system because $r | q^2 - 1$ and hence exponentiation with r kills all factors in μ_r. But we will see later that the condition $\delta_1, \delta_2 \in \mathcal{D}$ allows us to counteract this with additional oracle queries if $\mathcal{D}$ is sufficiently small.

Split exponents: After the previous step, only the exponents $q-1$ and $q+1$ occur in $\mathcal{F}$. Both are already light of weight 2 and we do not apply this step.

Transformation to the base field: Based on Section 5.3.6, we transform the functions in (5.27) and (5.28) into polynomials over $\mathbb{F}_q$. For $i \in \{0,1\}$, we define $\phi_i(x_{i,0}, x_{i,1}) = x_{i,0} + x_{i,1}\theta$. Because y_1 and y_2 represent $\delta_1, \delta_2 \in \mathbb{F}_q$ and following Remark 5.17 we define

$$\phi(x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1}, y_1, y_2) = (\phi_0(x_{0,0}, x_{0,1}), \phi_1(x_{1,0}, x_{1,1}), y_1, y_2).$$

With the q-ary representations $(-1, 1)$ and $(1, 1)$ of $q-1$ and $q+1$ and with $\phi_i^q(x_{i,0}, x_{i,1}) = x_{i,0} - x_{i,1}\theta$ we bring the functions in (5.27) and (5.28) into the form of (5.16) on page 73. After expanding products and sorting terms with respect to the basis $\{1, \theta\}$ of $\mathbb{F}_{q^2}/\mathbb{F}_q$ we obtain:

$$(x_{1,0} - 1)x_{0,0} + v x_{1,1} x_{0,1} + (x_{1,1} x_{0,0} + (x_{1,0} + 1) x_{0,1})\,\theta \tag{5.29}$$

$$x_{1,0}^2 - v x_{1,1}^2 - 1 \tag{5.30}$$

$$1 + 2x_{1,0} y_1 + y_1^2 - a_1^r \tag{5.31}$$

$$1 + 2x_{0,0} y_2 + (x_{0,0}^2 - v x_{0,1}^2) y_2^2 - a_2^r. \tag{5.32}$$

We are interested in $\mathbb{F}_q$-rational roots of this system. Because the exponentiation with $q+1$ is the norm from $\mathbb{F}_{q^2}$ to $\mathbb{F}_q$, (5.30)–(5.32) are defined over $\mathbb{F}_q$. Hence, we obtain two equations from (5.29) and three equations from (5.30)–(5.32). Furthermore, the determinant $(x_{1,0} - 1)(x_{1,0} + 1) - v x_{1,1}^2 = x_{1,0}^2 - v x_{1,1}^2 - 1$ of (5.29)

as linear system in $x_{0,0}, x_{0,1}$ equals (5.30). Hence, only one of the two equations defined by (5.29) brings new information and we remove the second equation $(x_{1,1}x_{0,0} + (x_{1,0}+1)x_{0,1})\theta = 0$. Finally we obtain the degree 4 model

$$(x_{1,0}-1)x_{0,0} + vx_{1,1}x_{0,1} \qquad 1 + 2x_{0,0}y_2 + (x_{0,0}^2 - vx_{0,1}^2)y_2^2 - a_2^r \quad (5.33)$$

$$x_{1,0}^2 - vx_{1,1}^2 - 1 \qquad 1 + 2x_{1,0}y_1 + y_1^2 - a_1^r. \quad (5.34)$$

Fix variables: The model defined by (5.33) and (5.34) is given by a system of four equations in six variables. If $\mathcal{D}$ is sufficiently small, we can guess values for the faults δ_1 and δ_2 in $\mathcal{D}$ and assign them to y_1 and y_2. This reduces the number of variables to 4.

Compute solutions: If we solve the system of (5.33) and (5.34) for each guess of δ_1 and δ_2, the complexity of the analysis is quadratic in $\#\mathcal{D}$. The authors of [LFG13] show how the complexity can be reduced. This is achieved by solving the system in two steps and verify partial solutions of the first step before computing complete solutions in the second step. Here, we benefit from the additional variable x_1 for $t = b^{q-1}$ in the initial model.

With (5.34) we obtain polynomials that are independent of $x_{0,0}$, $x_{0,1}$, and y_2. For each guess $\tilde{\delta}_1$ of δ_1 that we assign to y_1 we first solve (5.34) for $x_{1,0}$ and $x_{1,1}$. Because $x_{1,0}$ and $x_{1,1}$ represent t, we obtain candidates for t. For all candidates $\tilde{t}$, we check if the answer a_1 from the oracle is satisfied: i.e., we test that

$$\left(\tilde{t} + \tilde{\delta}_1\right)^{(q+1)/r} = a_1. \quad (5.35)$$

For each survivor $\tilde{t}$ of this test and each guess $\tilde{\delta}_2 \in \mathcal{D}$ that we assign to y_2, we then solve (5.33) for $x_{0,0}$ and $x_{0,1}$ to obtain candidate solutions for b. We see that this two-step approach reduces the number of guesses that we need to consider if only a few candidates survive the first test in (5.35).

Verify solutions: For each candidate $\tilde{b}$ for b from the previous step, we verify that this candidate satisfies the remaining answers from the oracle: i.e., we test if

$$f_0(\tilde{b}, 0) = a_0 \qquad \text{and} \qquad f_1(\tilde{b}, \tilde{\delta}_2) = a_2.$$

If this is the case, we add $\tilde{b}$ to the set $\mathcal{S}_{a_1,a_2}$ of surviving candidate solutions. Note that this set contains b.

Repetitions: So far, we showed how we can compute a set $\mathcal{S}_{a_1,a_2}$ of candidates for b. To reduce the number of solutions, we query the f_1 oracle for multiple times to obtain $a_3 = f_1(b, \delta_3), a_4 = f_1(b, \delta_4), \ldots$ with associated sets of solutions $\mathcal{S}_{a_1,a_3}, \mathcal{S}_{a_1,a_4}, \ldots$. Then we intersect all sets to obtain $\mathcal{S} = \bigcap_{i\geq 2} \mathcal{S}_{a_1,a_i}$ with $b \in \mathcal{S}$. In [LFG13] the authors show by simulating the attack that for $\#\mathcal{D} \leq 2^{10}$, already one additional query to the f_1 oracle with output a_3 is sufficient to reduce the number of candidates to $\#\mathcal{S} \leq 10$.

Discussion

This attack is an example where the exponent can be rescaled to an exponent of small weight. In particular, the exponent $(q+1)/r$ from (5.26) was rescaled to $q+1$ in (5.28). This enabled us to obtain small degree polynomials in (5.31) and (5.32) by applying Weil restriction of scalars. The crucial point is that the faults δ_1 and δ_2 modify the element $t = b^{q-1} \in \mu_{q+1}$ into an element with order not dividing $q+1$. Hence, the exponentiation with $q+1$ becomes non-trivial and we obtain non-trivial equations (5.31) and (5.32). We see that without faults, i.e., with $y_1 = y_2 = 0$, these equations are trivial after rescaling the exponent.

This attack is also an example where it is crucial to be able to guess assignments for the fault. Otherwise, (5.33) and (5.34) is under-determined and the corresponding algebraic set has positive dimension.

Chapter 6.

New fault attacks on pairings

In this chapter, we present two new attacks on Pair-Argument (cf. Definition 3.17) that we analyze based on our framework from Chapter 5: In Section 6.2 we extend our work from [Blö+14] where we presented the first practical second order attack on the complete pairing computation, including the Miller algorithm *and* the final exponentiation. In Section 6.3, as a new result of this thesis, we introduce an attack for the popular Ate pairing on BN curves.

In this chapter, we concentrate on the theoretical analysis of the attacks, i.e., we show how the secret argument of the pairing can be computed from the erroneous result of the pairing computation. In Chapter 8 and more specifically in Section 8.5, we explain how we realized the attack from [Blö+14] in practice in order to obtain the erroneous pairing result. We start with an overview of the related work and an outline of our contribution.

6.1. Introduction

For a general introduction on the techniques to analyze fault attacks on pairings we refer to Section 5.1. In this section, we present an overview of existing attacks and summarize our contribution.

6.1.1. Related work

The first theoretical fault attack on pairing computations was published in 2004 [PV04; PV06]. Since then, various other attacks were described and also performed in practice. There are already several surveys on the subject [Whe+09; MPV12; BGL14; Mra+15]. Here, we give an overview of the contributions that are most important from our point of view.

Page and Vercauteren [PV06] present a fault attack on the Eta pairing for fields $\mathbb{F}_q$ of characteristic $p = 2$ and $p = 3$. They tamper with the loop counter of Algorithm 3.4 such that either i or $i + 1$ iterations are computed. Then they show that two such outputs of Algorithm 3.4 can be used to separate a linear factor $f(P, Q)$ from $\mathrm{mil}_{r,P}(Q)$, and that the secret P or Q can be computed from the equation of this factor. With respect to the final exponentiation, the attack targets an implementation of the Eta pairing without a truncated Miller loop (cf. [Bar+07]). In this case, the final exponent is given as $q^p - 1$. Based on this special form, Page

and Vercauteren use the linearity of the q-th power Frobenius automorphism to invert the exponent without faults. For more details on the attack, we refer to Section 5.4.1.

Galbraith, hEigeartaigh, and Sheedy [GhS07] show that the Eta pairing without a truncated loop can be computed without final exponent. As in [PV06] the algebraic structure of the exponent $q^p - 1$ and the linearity of the Frobenius automorphism is used to remove the explicit computation of the final exponentiation. This underlines the result of [PV06] that shows that a final exponent with a special structure is not an obstacle for an attack.

Whelan and Scott [WS07] extend the ideas of [PV06] to other pairings like the Weil and the Tate pairing, as well as to other types of faults. They do not restrict to faults that modify the structure of the Miller algorithm as in [PV06]. Also faults that modify data like the coordinates of the points P, R, or Q in Algorithm 3.4 are considered. It is shown in [WS07] that these faults can also be used to separate a small degree factor $f(P,Q)$ from $\mathrm{mil}_{r,P}(Q)$. In the case where $f(P,Q)$ is linear, Whelan and Scott show how the secret can be computed from the equation of $f(P,Q)$. If the exponent has the special form $q-1$, a first order attack on the Weil pairing is described. In this attack, the linearity of the Frobenius automorphism is used to invert the final exponentiation without faults. Furthermore, it is argued in [WS07] that this attack does not apply to the Tate pairing where the final exponent is $\left(q^k - 1\right)/r$ and has not the required special structure. Hence, Whelan and Scott conclude that the final exponentiation can protect against first order fault attacks in many cases.

Mrabet [Mra09] presents an attack on the Miller algorithm that applies similar ideas as [PV06] and [WS07]. For the attack it is assumed that the loop of Algorithm 3.4 is terminated after i or $i+1$ iterations. As before, a factor $f(P,Q)$ of low degree can be separated from $\mathrm{mil}_{r,P}(Q)$ and then inverted to compute the secret P or Q. In addition to [WS07], the analysis is shown for Jacobian coordinates and also for non-linear factors $f(P,Q)$. With respect to the final exponentiation of the Tate pairing, it is assumed that it can be removed by other means. Mrabet proposes to read out the result of the Miller algorithm in test mode by means of a scan attack. In [Mra10] the analysis of [Mra09] is extended to curves in Edwards coordinates. As in [Mra09] it is assumed that the final exponentiation can be eliminated by other means.

Bae, Moon, and Ha [BMH13] describe also an attack on the Miller algorithm. For the attack, they assume that the `if` branch in the last iteration of Algorithm 3.4 is skipped. For points P and Q of odd order r, this allows to isolate a linear function from $\mathrm{mil}_{r,P}(Q)$. Then the equation of this function can be solved for P or Q. To demonstrate the feasibility of their attack, Bae, Moon, and Ha use a laser pulse to skip a branch instruction on an AVR ATmega128L. In their practical evaluation, they do not attack a complete pairing computation. Instead, they attack a small test program including a conditional branch and a trigger for the laser that sends a special output in case the fault injection was successful. As in the previous attacks, it is assumed that the final exponentiation is inverted by other means.

So far, we described attacks that did not target the final exponentiation with faults. To invert the final exponentiation, either exponents with a special structure were considered, e.g., in [PV06], or it was assumed that the exponentiation can be inverted by other means [Mra09]. Lashermes, Fournier, and Goubin [LFG13] go in the opposite direction. They present an attack on the final exponentiation and show how the output of the Miller algorithm can be recovered if faults are introduced into elements in $\mathbb{F}_{q^k}$ during the exponentiation with $\left(q^k - 1\right)/r$. Inversion of the Miller algorithm is not considered. The attack requires three pairing computations where one fault is introduced at each exponentiation. Lashermes, Fournier, and Goubin simulate the analysis of the attack and show that in the case of precise faults that effect less than 10 bits of an element in $\mathbb{F}_{q^k}$, the exponentiation can be reversed up to a few candidate solutions. For more details on the attack, we refer to Section 5.4.2.

Lashermes et al. [Las+14] practically validate previous theoretical attacks on the Miller algorithm. They analyze faults in elements of $\mathbb{F}_q$ as well as faults that tamper with the number of iterations of Algorithm 3.4. The practical evaluation is performed on an ARM Cortex M3 processor and EM pulses are used as the fault injection technique. The concrete implementation is a C implementation of the Ate pairing from the authors of the attack. The attack concentrates on the Miller algorithm and assumes that the final exponentiation can be removed by other means. The authors leave it as an open problem to combine their attack with an attack on the final exponentiation like in [LFG13], to obtain a practical second order attack on the complete pairing. Lashermes et al. also show that randomized projective coordinates are not effective for fault attacks on the Miller algorithm.

Point blinding has been proposed as a countermeasure, especially against passive attacks on the Miller algorithm [PV06]. Here, the secret or the public argument of the pairing is masked by adding a random point. Park, Sohn, and Moon [PSM11] show that this countermeasure is not always effective. They exploit the fact that the Eta pairing in characteristic 3 is defined over $\mathbb{F}_{q^6}$, while the secret and the point used for randomization are already defined $\mathbb{F}_q$ (cf. Remark 5.17). This results in a system of equations that can be solved for the secret as well as for the mask. The approach of [PSM11] was generalized in [MF15] and applied also to other pairings.

6.1.2. Our contribution

This chapter contains two major contributions:

1. In Section 6.2 we analyze and extend our attack from [Blö+14].
2. In Section 6.3 we present and analyze a new fault attack on the complete Ate pairing that we did not publish so far.

For both attacks, we use a similar new technique to target the final exponentiation. With a fault, we skip an extension field multiplication during the final exponentiation. This virtually introduces an additive term $\delta \in \mathbb{Z}$ into the second factor of the final

Pairing	FE	Target	Pract. eval.	Reference
Eta	q^2-1, q^3-1	Miller	no	[PV04; PV06]
Eta	1	Miller	no	[WS07]
Weil	$q-1$	Miller	no	[WS07]
Weil	1	Miller	no	[Mra09; Mra10]
Tate/Ate	$\left(q^k-1\right)/r$	Miller	no	[WS07; Mra09; Mra10]
Tate/Ate	$\left(q^k-1\right)/r$	FE	no	[LFG13]
Tate/Ate	$\left(q^k-1\right)/r$	Miller	yes	[BMH13; Las+14]
Eta	$q-\sqrt{2q}+1$	Miller+FE	yes	[Blö+14], Section 6.2
Tate/Ate	$\left(q^k-1\right)/r$	Miller+FE	no	Section 6.3

Table 6.1.: Summary of related work and our contribution with respect to fault attacks on pairings. The column FE lists the final exponent of the targeted pairing. Here, q is the size of the base field, k is the embedding degree, and r is the order of the pairing groups. Most previous attacks were not evaluated in practice and consider only faults at the Miller algorithm. We remark that for non-trivial exponents $\left(q^k-1\right)/r$ or $q-\sqrt{2q}+1$, faults at the Miller algorithm and at the final exponentiation are required to invert the complete pairing (cf. Section 5.1.1).

exponent (cf. Section 3.2.3) such that it is modified as follows:

$$\frac{(q^k-1)}{\Phi_k(q)}\frac{\Phi_k(q)}{r} \rightsquigarrow \text{fault} \rightsquigarrow \quad \frac{(q^k-1)}{\Phi_k(q)}\left(\frac{\Phi_k(q)}{r}+\delta\right). \tag{6.1}$$

Then we show how we can invert this exponent based on two algebraic properties:

1. The small weight of the factor $(q^k-1)/\Phi_k(q)$ in base q (cf. Definition 2.1).
2. A small greatest common divisor of $\Phi_k(q)/r+\delta$ and the order q^k-1 of $\mathbb{F}_{q^k}^*$.

We give more details on both contributions in the following. Furthermore, Table 6.1 compares our contribution with the related work.

Extension of our attack from [Blö+14] on the Eta pairing

At FDTC 2014 [TC14], we presented the first practical realization of a fault attack on a complete pairing computation, including the Miller algorithm and the final exponentiation [Blö+14]. Hence, we answered the question if it is possible to practically attack the Miller algorithm and the final exponentiation simultaneously which Lashermes et al. pose in [Las+14] also at FDTC 2014:

> "[...] how to properly override the final exponentiation in conjunction with a fault attack on the Miller algorithm remains an open problem which has to be further studied, [...]

> A complete fault attack on a pairing taking into account the final exponentiation would be of interest both theoretically and experimentally."

As in the attack from [PV06], we attacked the Eta pairing (see also [Bar+07] and Section 5.4.1). For the concrete instantiation, we used the implementation of the RELIC library [AG]. Compared to purely theoretical attacks like the one in [PV06] we had to master additional difficulties with influence on the theoretical analysis of the attack:

- Our target implementation of the Eta pairing used loop unrolling for efficiency reasons. Hence, even though we terminated the loop of the Miller algorithm (see Algorithm 3.4) after one iteration, we are not able to isolate a single factor like (5.18) from page 76 as in the attack of [PV06]. This has two consequences:

  1. We cannot exploit the special form of (5.18) as it was done in [PV06] to invert the final exponentiation.
  2. The complexity of the analysis is increased compared to the attack from [PV06].

  Based on our framework from Chapter 5 we show in Section 6.2 how we deal with this problem.

- Compared to [PV06], our target implements the more efficient version of the Eta pairing with truncated loop length (cf. Section 6.1 of [Bar+07]). In this case, the final exponent has not the special form $q^p - 1$ with small weight w_q that allowed the authors of [PV06] to invert the exponentiation without faults.

  To deal with this problem, in [Blö+14], we use a second order attack to remove the exponentiation with the second fault. More concretely, we used an instruction skip fault to skip the `call` instruction to the final exponentiation (cf. Section 8.5).

- Sometimes it is not possible to skip the complete final exponentiation with the second fault. We observed, for example, that the compiler inlines the code of the final exponentiation at higher optimization levels. Hence, we cannot skip the `call` instruction anymore (cf. Section 8.5.4).

  In Section 6.2 we extend our work from [Blö+14] and show how we can solve this problem. In our extension, we do not assume that the final exponentiation is completely removed. Instead, we first show how the exponent can be modified as in (6.1) by skipping an extension field multiplication. Then we show that the modified exponent can be inverted based on our framework from Chapter 5. The extended attack on the final exponentiation is a new contribution of this thesis that we did not publish so far.

New attack on the Ate pairing

In Section 6.3 we transfer the basic concepts of our extended attack on the final exponentiation from Section 6.2 to the optimal Ate pairing [Ver10] on BN curves — one of the most popular settings for PBC today. This result is a new contribution of this thesis that we did not publish so far.

We use two instruction skips to attack the Ate pairing. With the first instruction skip, we isolate a line (cf. Definition 3.8) from the Miller function of Definition 2.31. With the second fault we skip one multiplication within the final exponentiation (cf. Section 3.2.3) to modify the exponent as in (6.1).

Based on our framework from Chapter 5 we show how the secret can be computed from two erroneous pairing computations. In the analysis, we exploit the fact that the arguments of the Ate pairing for BN curves are defined over a proper subfield of $\mathbb{F}_{q^k}$ (see Remark 5.17). Furthermore, we use the properties of the erroneous final exponent that we described above. We show how this allows us to describe the secret by polynomial equations of relatively small degree. To verify the efficiency of the analysis, we simulated our attack for parameters at the 128 bit security level.

6.2. Attack on the Eta pairing

In this section, we analyze an extension of our second order attack from [Blö+14]. In [Blö+14] we used the second of our two faults to completely skip the final exponentiation. Here, we present an attack where the final exponentiation is not completely skipped. Instead, we use the second fault to modify the exponent such that it can be inverted in the analysis. We note that the analysis of the basic attack from [Blö+14] without exponentiation follows from the analysis of this section by leaving out all steps that are related to the exponent. Here, we concentrate on the theoretical analysis of the attack. In Section 8.5 we give details about the practical realization of the attack and we also give a practical motivation of our extension from this section.

6.2.1. Background information on the attacked implementation

We attack an implementation of the Eta pairing [Bar+07] in characteristic 2 on supersingular elliptic curves. Now we recall the most important parameters of the pairing from Section 6 and especially Section 6.1 of [Bar+07]. The Eta pairing is defined based on the elliptic curve $E : y^2 + y = x^3 + x$ over the finite field $\mathbb{F}_q$ with $q = 2^m$. For our concrete target implementation m is set to $m = 271$. For our case, i.e., $m = 7 \mod 8$, it holds that $\#E(\mathbb{F}_q) = 2^m + 2^{(m+1)/2} + 1$. The embedding degree of q and $\#E(\mathbb{F}_q)$ is $k = 4$. We define the extension field $\mathbb{F}_{q^4} = \mathbb{F}_q(s, t)$ with $s^2 = s + 1$ and $t^2 = t + s$. Let $e = (q^4 - 1)/\#E(\mathbb{F}_q) = (2^{2m} - 1)(2^m - 2^{(m+1)/2} + 1)$ and $n = 2^{(m+1)/2} + 1$. Define the map $\psi(x, y) = (x + s + 1, y + sx + t)$. For inputs

$P, Q \in E(\mathbb{F}_q)$ the Eta pairing $\eta : E(\mathbb{F}_q) \times E(\mathbb{F}_q) \to \mathbb{F}_{q^k}^*$ is then defined as

$$\eta(P, Q) = \mathrm{mil}_{n,-P}(\psi(Q))^e.$$

Algorithm 6.1 Implementation of $\mathrm{mil}_{n,-P}(\psi(Q))$ on $E(\mathbb{F}_{2^m})$ for $m = 7 \mod 8$ and $E : y^2 + y = x^3 + x$.

Input: $P = (x_P, y_P), Q = (x_Q, y_Q) \in E(\mathbb{F}_{2^m})$
Output: $\mathrm{mil}_{n,-P}(\psi(Q))$
1: $u \leftarrow x_P$, $v \leftarrow x_Q$
2: $T \leftarrow u \cdot v + y_P + y_Q + (u + x_Q)s + t$
3: $u \leftarrow x_P^2$
4: $L \leftarrow T + v + u + s$
5: $b \leftarrow T \cdot L$
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: $\quad T \leftarrow u \cdot v + y_P + y_Q + (u + x_Q)s + t$
11: $\quad b \leftarrow b \cdot T$
12: **end for**
13: **return** b

Our target implementation is similar to Algorithm 3 of [Bar+07] but adapted to the parametrization $m = 7 \mod 8$. It is presented in Algorithm 6.1. Different from Algorithm 3 of [Bar+07] the first loop is unrolled and computed in Line 3–Line 5 of Algorithm 6.1 before the computation of the main loop. Because we attack the exponentiation with $e = \left(q^2 - 1\right)\left(q - \sqrt{2q} + 1\right)$, we list the target implementation of this exponentiation in Algorithm 6.2.

6.2.2. Background information on the attack

With the first fault, we terminate the loop of Algorithm 6.1 after the execution of the first iteration. With the second fault, we skip the multiplication in Line 6 of Algorithm 6.2. This results in the modified exponent $\tilde{e} = \left(q^2 - 1\right)\left(-\sqrt{2q} + 1\right)$.

Define

$$\begin{aligned} T(x_1, y_1; x_2, y_2) &= x_1 x_2 + y_1 + y_2 + (x_1 + x_2)s + t \\ L(x_1, y_1; x_2, y_2) &= T(x_1, y_1; x_2, y_2) + x_1^2 + x_2 + s. \end{aligned} \tag{6.2}$$

From Algorithm 6.1 we see that the attack implements an oracle for the function

$$g(x_1, y_1; x_2, y_2) = \left(T\left(x_1, y_1; x_2, y_2\right) L\left(x_1, y_1; x_2, y_2\right) T\left(\sqrt{x_1}, \sqrt{y_1}; x_2^2, y_2^2\right)\right)^{\tilde{e}}. \tag{6.3}$$

Let a be the result of the erroneous pairing computation on input P and Q, i.e., $a = g(P, Q)$.

Algorithm 6.2 Implementation of exponentiation with $e = \left(q^2-1\right)\left(q-\sqrt{2q}+1\right)$

Input: $b \in \mathbb{F}_{q^4}$ with $q = 2^m$, $e = \left(q^2-1\right)\left(q-\sqrt{2q}+1\right)$
Output: b^e

1: $v_1 \leftarrow \sigma_{q^2}(b)$ ▷ $v_1 = b^{q^2}$
2: $v_2 \leftarrow b^{-1}$ ▷ $v_2 = b^{-1}$
3: $v_2 \leftarrow v_1 v_2$ ▷ $v_2 = b^{q^2-1}$
4: $v_3 \leftarrow v_2$ ▷ $v_3 = b^{q^2-1}$
5: $v_1 \leftarrow \sigma_q(v_2)$ ▷ $v_1 = b^{(q^2-1)q}$
6: $v_2 \leftarrow v_1 v_2$ ▷ $v_2 = b^{(q^2-1)(q+1)}$
7: **for** $i \leftarrow 1 \ldots (m+1)/2$ **do** ▷ Exponentiation with $\sqrt{2q} = 2^{(m+1)/2}$
8: $v_3 \leftarrow v_3^2$ ▷ $v_3 = b^{(q^2-1)2^i}$
9: **end for**
10: $v_3 \leftarrow v_3^{-1}$ ▷ $v_3 = b^{-(q^2-1)\sqrt{2q}}$
11: $v_1 \leftarrow v_2 v_3$ ▷ $v_1 = b^{\left(q^2-1\right)\left(q-\sqrt{2q}+1\right)}$
12: **return** v_1

6.2.3. Analysis of the attack

In this section, we analyze the attack based on the approach from Section 5.2. We leave out the steps that do not apply to this attack. Our analysis proves the following theorem that summarizes the efficiency of the attack:

Theorem 6.1. *Let P and Q be defined as in Section 6.2.2 and $S \in \{P, Q\}$. From $a = g(P, Q)$ we can derive a set of bivariate polynomial systems $\mathcal{F}_1, \ldots, \mathcal{F}_5$ such that one of the systems is a model for S. The degree of $\mathcal{F}_1, \ldots, \mathcal{F}_5$ is bounded by 7 for $S = P$ and by 4 for $S = Q$.*

From Definition 5.6 we see that we can compute candidates for the secret argument S of the pairing by solving at most five bivariate equation systems and searching trough the resulting candidate solutions. We also see that for $S = P$ the analysis is potentially more complex than for $S = Q$ because of the higher degree of the model.

Initial model

We define the model based on (6.3). But to eliminate $\sqrt{x_1}$ and $\sqrt{y_1}$, we substitute x_1 with x_1^2, y_1 with y_1^2, $\sqrt{x_1}$ with x_1, and $\sqrt{y_1}$ with y_1 and define

$$f(x_1, y_1; x_2, y_2) = T\left(x_1^2, y_1^2; x_2, y_2\right) L\left(x_1^2, y_1^2; x_2, y_2\right) T\left(x_1, y_1, x_2^2, y_2^2\right). \tag{6.4}$$

We obtain a model for P and Q:

Lemma 6.2. *The function*

$$f(x_1, y_1; x_2, y_2)^{\tilde{e}} - a \tag{6.5}$$

defines a model for P and Q.

Proof. By inserting $\left(\sqrt{x(P)}, \sqrt{y(P)}; x(Q), y(Q)\right)$ into $f(x_1, y_1; x_2, y_2)$ from (6.4) it directly follows that

$$f\left(\sqrt{x(P)}, \sqrt{y(P)}; x(Q), y(Q)\right)^{\tilde{e}} = g(P, Q) = a$$

and hence, $\left(\sqrt{x(P)}, \sqrt{y(P)}; x(Q), y(Q)\right)$ is a root of (6.5). Furthermore, we can efficiently compute both, P or Q, from this root. Hence, the claim follows from Definition 5.6. □

Combine exponents

We apply Lemma 5.13 with $e_1 = \left(q^2 - 1\right)\left(-\sqrt{2q} + 1\right)$ to remove the factor of e_1 that is coprime to $q^4 - 1$. It holds that

$$\gcd\left(q^4 - 1, \left(q^2 - 1\right)\left(-\sqrt{2q} + 1\right)\right) = (q^2 - 1)\gcd(q^2 + 1, -\sqrt{2q} + 1).$$

For our choice of $q = 2^{271}$ we obtain $\gcd(q^2 + 1, -\sqrt{2q} + 1) = 5$. We use the EEA to compute $u_0, u_1 \in \mathbb{Z}$ with

$$(q^2 - 1)5 = u_0(q^4 - 1) + u_1\left(q^2 - 1\right)\left(-\sqrt{2q} + 1\right).$$

Then we replace (6.5) with

$$f(x_1, y_1; x_2, y_2)^{(q^2-1)5} - a^{u_1}$$

and obtain a new model for S.

Split exponents

We split $(q^2 - 1)5$ into the light factor $l = (q^2 - 1)$ with $\mathrm{w}_q(l) = 2$ and the heavy factor $h = 5$. Furthermore, we invert $h = 5$ and compute $b_1, \ldots, b_5$ with $b_i^5 = a^{u_1}$. Because $5 \parallel q^4 - 1$ this can be done efficiently by computing

$$\{b_1, \ldots, b_5\} = a^{u_1\left(5^{-1} \bmod (q^4-1)/5\right)}\mu_5. \tag{6.6}$$

Then we continue with five different models $\mathcal{F}_1, \ldots, \mathcal{F}_5$, where the i-th model is defined by the equation

$$f(x_1, y_1; x_2, y_2)^{q^2-1} - b_i. \tag{6.7}$$

Transformation to the base field

Now we transform the models into polynomial systems over $\mathbb{F}_q$. With $P, Q \in E(\mathbb{F}_q)$ and according to Remark 5.17 we define $\phi(x_1, y_1; x_2, y_2) = (x_1, y_1; x_2, y_2)$ to apply

the Weil restriction of scalars. The q-ary representation of q^2-1 from Definition 2.1 is $(\epsilon_0, \epsilon_1, \epsilon_2) = (-1, 0, 1)$. Hence, from (6.7) we obtain

$$f^{q^2}(x_1, y_1; x_2, y_2) - b_i f(x_1, y_1; x_2, y_2) \tag{6.8}$$

for (5.16) from page 73. We see that the degree of (6.8) is bounded by the degree of $f(x_1, y_1; x_2, y_2)$.

For each b_i, we sort terms with respect to the basis $\{1, \sigma, \rho, \sigma\rho\}$. For every b_i, together with the Weierstrass equation of the curve, we obtain a system of 5 polynomial equations defined over $\mathbb{F}_q$. The system with the correct b_i that fulfills

$$b_i = f\left(\sqrt{x(P)}, \sqrt{y(P)}; x(Q), y(Q)\right)^{q^2-1}$$

is a model for S.

Fix variables

For the case $S = P$ we assign $(x_2, y_2) = (x(Q), y(Q))$, and for the case $S = Q$, we assign $(x_1, y_1) = \left(\sqrt{x(P)}, \sqrt{y(P)}\right)$. Then we continue with polynomial equations in $\mathbb{F}_q[x_1, y_1]$ or $\mathbb{F}_q[x_2, x_2]$, respectively. From (6.2) and (6.4) we see that the degree of (6.8) after fixing variables is bounded by 7 for $S = P$ and by 4 for $S = Q$. This completes the proof of Theorem 6.1.

Compute and verify solutions

In this step we enumerate $X = \bigcup_{i=1}^{5} V(\mathcal{F}_i)(\mathbb{F}_q)$ with standard algorithms. For $S = Q$, the elements in X are candidate solutions. We see from the proof of Theorem 6.1 that for $S = P$, and $(\alpha, \beta) \in X$ a candidate for P is given by (α^2, β^2). Finally, we verify each candidate based on a correct pairing computation to identify S.

6.2.4. Discussion

Now we compare the efficiency of the analysis with the efficiency of our original attack from [Blö+14] where the final exponent was completely removed. On the one hand, we see from (6.8) that the degree of the final model is not increased by the factor $q^2 - 1$ of the exponent. On the other hand, due to the heavy factor 5 of the exponent we generated five systems based on $b_1, \ldots, b_5$ from (6.6) that we need to solve. Hence, the overall complexity of the analysis is increased by a factor of 5 compared to our attack from [Blö+14].

6.2.5. Simulation based evaluation of the efficiency

We implemented the analysis with Sage [Sag14]. We computed the candidates based on the `variety()` function that uses Groebner bases techniques. We simulated 30

runs of the attack for the more complex case where $S = P$ and and we were able to compute the secret from an erroneous pairing computation in less than 30 seconds on a standard PC.

We also used a similar analysis in the practical realization of our basic attack from [Blö+14], where the exponent was completely removed and where the secret was $S = Q$. There, and consistent with our discussion in Section 6.2.4, the computation of the secret from an erroneous pairing result took only a few seconds.

We see that in both cases, the analysis is very efficient. In practice an efficient analysis is crucial. The reason is that in a practical attack, typically many experiments have to be performed until the intended fault is injected (cf. Section 8.5) and because we can distinguish a successful experiment from a failed experiment only by verifying the solutions of the analysis.

6.3. Attack on the Ate pairing over BN curves

In this section, we outline an attack on the optimal Ate pairing [Ver10] over BN curves (cf. Theorem 3.6). It exploits that BN curves have a twist of degree 6. This allows us to reduce the number of variables in the model that we use as input for the computation of the secret argument of the pairing. Furthermore, we introduce a fault in the final exponentiation that allows us to invert the exponentiation with the erroneous exponent.

6.3.1. Background information on the attacked implementation

The optimal Ate pairing is an optimization of the reduced Ate pairing from Definition 2.38. As for the reduced Ate pairing, the optimal Ate pairing relies on the fact that with $P \in \mathbb{G}_2$ and $Q \in \mathbb{G}_1$ (see Definition 2.18) both arguments are elements of the eigenspaces of π_q.

Recall from Theorem 3.6 that a BN curve E is defined over a prime field $\mathbb{F}_q$, has prime order $\#E(\mathbb{F}_q) = r$, and the embedding degree of q and r is $k = 12$. With u from Theorem 3.6 and $L_{P_1,P_2}(x, y)$ from Definition 3.8 define $\lambda = 6u + 2$, $P_i = \pi_{q^i}(P)$, and

$$M_P(Q) = L_{P_3,-P_2}(Q)L_{P_3-P_2,P_1}(Q)L_{P_1-P_2+P_3,\lambda P}(Q).$$

Over a BN curve the optimal Ate pairing can be defined as follows (see Section IV of [Ver10]):

$$\begin{aligned} \text{ate}_{\text{BN}} : \mathbb{G}_2 \times \mathbb{G}_1 &\to \mu_r \\ (P, Q) &\mapsto \left(\text{mil}_{\lambda,P}(Q)M_P(Q)\right)^{\left(q^k-1\right)/r}. \end{aligned} \tag{6.9}$$

We assume a straight-forward implementation for the computation of $\text{mil}_{\lambda,P}(Q)$ and $M_P(Q)$. We further assume that the implementation of the final exponentiation with $\left(q^k - 1\right)/r$ is computed in two steps as described in Section 3.2.3. For our

Algorithm 6.3 Outline of algorithm for computing the Ate pairing on BN curves

Input: $P \in \mathbb{G}_2$, $Q \in \mathbb{G}_1$, $\lambda \in \mathbb{Z}$, $e = (q^4 - q^2 + 1)/r$ with binary representation $e = \sum_{j=0}^{n} e_j 2^j$
Output: $\mathrm{ate}_{\mathrm{BN}}(P, Q)$

1: $b \leftarrow \mathrm{mil}_{\lambda,P}(Q)$
2: $P_1 \leftarrow \pi_q(P)$
3: $P_2 \leftarrow \pi_q(P_1)$
4: $P_3 \leftarrow \pi_q(P_2)$
5: $b \leftarrow b \cdot L_{P_3,-P_2}(Q)$
6: $b \leftarrow b \cdot L_{P_3-P_2,P_1}(Q)$
7: $b \leftarrow b \cdot L_{P_1-P_2+P_3,\lambda P}(Q)$
8: $b \leftarrow b^{q^8+q^6-q^2-1}$
9: $a \leftarrow b$
10: **for** $j \leftarrow n-1 \ldots 0$ **do**
11: $\quad a \leftarrow a^2$
12: $\quad$ **if** $e_j = 1$ **then**
13: $\quad\quad a \leftarrow a \cdot b$
14: $\quad$ **end if**
15: **end for**
16: **return** a

case with $k = 12$ and $\Phi_{12}(q) = q^4 - q^2 + 1$ we obtain

$$\frac{q^{12}-1}{r} = \left(q^8 + q^6 - q^2 - 1\right) \frac{q^4 - q^2 + 1}{r}.$$

For simplicity, we assume that the exponentiation with the second factor is performed by a standard square and multiply approach. The basic structure of the complete pairing computation is outlined in Algorithm 6.3. There, we expanded the exponentiation with $e = \left(q^4 - q^2 + 1\right)/r$ because we target this computation in our attack.

6.3.2. Background information on the attack

We assume a second order attack. The first fault is applied only optionally. If it is applied, the multiplication in Line 5 of Algorithm 6.3 is skipped. The second fault is always applied. It skips the multiplication in Line 13 for one bit $e_j = 1$ of the exponent $e = \sum_{j=0}^{n} e_j 2^j$. Hence, we modify the final exponent into

$$\tilde{e} = \left(q^8 + q^6 - q^2 - 1\right)\left(\left(q^4 - q^2 + 1\right)/r - 2^j\right). \tag{6.10}$$

According to our notation from Definition 5.7, we re-write $\mathrm{mil}_{\lambda,P}(Q) = \mathrm{mil}_{\lambda}(P, Q)$, $L_{P_3,-P_2}(Q) = L(P_3, -P_2, Q)$, and $M_P(Q) = M(P, Q)$. With the definition of $P_i = \pi_{q^i}(P)$, we see from (6.9) that this attack implements an oracle for the

functions

$$\begin{aligned} f_1(x_1, y_1; x_2, y_2) &= \left(\mathrm{mil}_\lambda(x_1, y_1; x_2, y_2) M(x_1, y_1; x_2, y_2)\right)^{\tilde{e}} \\ f_2(x_1, y_1; x_2, y_2) &= \frac{f_1(x_1, y_1; x_2, y_2)}{L\left(\pi_{q^3}(\psi(x_1, y_1)); -\pi_{q^2}(\psi(x_1, y_1)); x_2, y_2\right)^{\tilde{e}}}. \end{aligned} \tag{6.11}$$

In the attack, we compute two erroneous pairings, one for each mode of the attack to obtain $a_1 = f_1(P, Q)$ and $a_2 = f_2(P, Q)$.

We remark that we explain in Section 8.1.3 how the attack could be realized in practice by means of instruction skips.

6.3.3. Analysis of the attack

In this section, we analyze the attack based on the approach of Section 5.2. We leave out the steps that do not apply to this attack. Our analysis will prove the following theorem that summarizes the efficiency of the attack:

Theorem 6.3. *Let P, Q, a_1, a_2, and j be defined as in Section 6.3.2. Furthermore, assume*

$$r \nmid \left(\left(q^4 - q^2 + 1\right)/r - 2^j\right).$$

From a_1 and a_2 we can derive a degree-8 model in $\mathbb{F}_q[x_{1,1}, x_{1,2}, y_{1,1}, y_{1,2}]$ for P and a degree-4 model in $\mathbb{F}_q[x_2, y_2]$ for Q.

We prove the theorem by performing the analysis of Section 5.2.2.

Initial model

From Corollary 3.7 we know that BN curves have a twist E' of degree 6. We define an initial model that later allows us to exploit that $\mathbb{G}_2$ has a very compact representation in this case (cf. Theorem 2.21 and Remark 2.22). Therefore, let $\psi : E' \to E$ be the isomorphism between E' and E. We define

$$f(x_1, y_1; x_2, y_2) = L\left(\pi_{q^3}(\psi(x_1, y_1)); -\pi_{q^2}(\psi(x_1, y_1)); x_2, y_2\right) \tag{6.12}$$

and obtain a model for P and Q:

Lemma 6.4. *Let $a = a_1/a_2$. Then the function*

$$f(x_1, y_1; x_2, y_2)^{\tilde{e}} - a \tag{6.13}$$

defines a model for P and Q.

Proof. Let $P' \in E'$ be the preimage of P under ψ. We claim that (P', Q) is a zero of (6.13). From (6.11) we see that

$$f(x_1, y_1; x_2, y_2)^{\tilde{e}} = \frac{f_1\left(\psi(x_1, y_1); x_2, y_2\right)}{f_2\left(\psi(x_1, y_1); x_2, y_2\right)}.$$

Then the claim follows from the definition of a.

Furthermore, we can efficiently compute P from P' based on ψ. Hence, (6.13) is a model for P and Q. □

Combine exponents

We apply Lemma 5.13 with $e_1 = \tilde{e}$ from (6.10) to remove the factor of $\tilde{e}$ that is coprime to $q^{12} - 1$. The following lemma shows that under a weak assumption on j, q, and r, we obtain a light exponent (cf. Definition 2.1) after this step.

Lemma 6.5. *Define $\tilde{e}$ as in* (6.10). *Let $j \geq 0$ such that $r \nmid \left(\left(q^4 - q^2 + 1\right)/r - 2^j\right)$. Then it holds that* $\gcd\left(q^{12} - 1, \tilde{e}\right) = q^8 + q^6 - q^2 - 1$.

Proof. First note that with r prime and on condition that $r \nmid \left(\left(q^4 - q^2 + 1\right)/r - 2^j\right)$ it holds that

$$\gcd(q^4 - q^2 + 1, \left(q^4 - q^2 + 1\right)/r - 2^j) = \gcd(q^4 - q^2 + 1, 2^j).$$

Because q is an odd prime $q^4 - q^2 + 1$ is also odd. Hence $\gcd(q^4 - q^2 + 1, 2^j) = 1$. Because $q^8 + q^6 - q^2 - 1$ divides both, $\tilde{e}$ and $q^{12} - 1$, the claim follows. □

From now on, we assume that the condition $r \nmid \left(\left(q^4 - q^2 + 1\right)/r - 2^j\right)$ of Lemma 6.5 is fulfilled. As in Section 5.3.3 we use the EEA to compute $u_0, u_1 \in \mathbb{Z}$ with

$$q^8 + q^6 - q^2 - 1 = u_0(q^{12} - 1) + u_1\tilde{e}.$$

Then we replace (6.13) with

$$f(x_1, y_1; x_2, y_2)^{\left(q^8+q^6-q^2-1\right)} - a^{u_1} \tag{6.14}$$

and obtain a new model for S.

Split exponents

Because the exponent from (6.14) has already small weight $\mathrm{w}_q\left(q^8 + q^6 - q^2 - 1\right) = 4$ we do not apply this step.

Transformation to the base field

Now we transform (6.14) into a polynomial system over $\mathbb{F}_q$. We see from Remark 2.22 that ψ induces an isomorphism from $E'(\mathbb{F}_{q^2})[r]$ to $\mathbb{G}_2$. Hence, for P' with $\psi(P') = P \in \mathbb{G}_2$ it holds that $P' \in E'(\mathbb{F}_{q^2})[r]$. Furthermore, with $Q \in \mathbb{G}_1$ the point Q is already defined over $\mathbb{F}_q$. Following Remark 5.17, we need to introduce only four variables $x_{1,1}$, $x_{1,2}$, and $y_{1,1}$, $y_{1,2}$ for the coordinates of P' to define the Weil restriction of scalars from Definition 2.2. Let $\mathbb{F}_{q^2} = \mathbb{F}_q(\theta)$ and $\mathbb{F}_{q^{12}} = \mathbb{F}_{q^2}(\alpha) = \mathbb{F}_q(\theta, \alpha)$. We abbreviate $\underline{x} = x_{1,1}, x_{1,2}, y_{1,1}, y_{1,2}; x_2, y_2$ and define

$$\begin{aligned} \phi : \mathbb{A}^6 &\to \mathbb{A}^4 \\ (\underline{x}) &\mapsto (x_{1,1} + x_{1,2}\theta, y_{1,1} + y_{1,2}\theta; x_2, y_2)\,. \end{aligned}$$

We apply the q^i-th power Frobenius maps to the coefficients of $f \circ \phi$ with f from (6.12). First note that the function L from Definition 3.8 has coefficients in $\mathbb{F}_q$ and

hence, $L^{q^i}(\,\cdot\,) = L(\,\cdot\,)$. Furthermore, with $\theta \in \mathbb{F}_{q^2}$, and hence $\theta^{q^i} = \theta^{q^i \bmod 2}$, we obtain

$$f^{q^i}\left(\phi^{q^i}(\underline{x})\right) = L\left(\psi^{q^{i+3}}\left(x_{1,1} + x_{1,2}\theta^{q^{i+1}}, y_{1,1} + y_{1,2}\theta^{q^{i+1}}\right), \right. \\ \left. -\psi^{q^{i+2}}\left(x_{1,1} + x_{1,2}\theta^{q^i}, y_{1,1} + y_{1,2}\theta^{q^i}\right); x_2, y_2\right). \quad (6.15)$$

We omit the details here, but now we clear denominators of L and use the representation

$$(\epsilon_0, \ldots, \epsilon_8) = (-1, 0, -1, 0, 0, 0, 1, 0, 1)$$

of $q^8 + q^6 - q^2 - 1$ in base q to write down (5.16) from page 73 for our case. As in (5.17) we then sort terms with respect to the basis $\left\{\theta^i\alpha^j\right\}_{0\le i<2, 0\le j<6}$ of $\mathbb{F}_{q^k}$ as a $\mathbb{F}_q$ vector space.

In summary, we transformed (6.14) into a system of $k = 12$ polynomial equations in $\mathbb{F}_q[\underline{x}]$. Finally, we also apply ϕ to the variables of the Weierstrass equations of E' and E to add them as additional constraints to the model.

Fix variables

For the case where P is secret, we fix $(x_2, y_2) = (x(Q), y(Q))$ to obtain a model in $\mathbb{F}_q[x_{1,1}, x_{1,2}, y_{1,1}, y_{1,2}]$. If the secret is Q, we write $P' \in E'(\mathbb{F}_{q^2})$ as $P' = (x_{1,1} + x_{1,2}\theta, y_{1,1} + y_{1,2}\theta)$ and fix variables accordingly. This provides a model in $\mathbb{F}_q[x_2, y_2]$.

Because ψ is an isomorphism of curves it has degree 1. For example, it could be defined as in (3.1) on page 28. Therefore the degree of (6.15) equals the degree of $L(\,\cdot\,)$. It follows from Definition 3.8 that the degree of (6.15) is bounded by 2 as polynomial in $\mathbb{F}_q[x_{1,1}, x_{1,2}, y_{1,1}, y_{1,2}]$. Similarly, the degree of (6.15) is bounded by 1 as polynomial in $\mathbb{F}_q[x_2, y_2]$.

It follows from Theorem 5.16 and $\mathrm{w}_q\left(q^8 + q^6 - q^2 - 1\right) = 4$ that the degree of the final model is bounded by 8 for secret P and by 4 for secret Q. This completes the proof of Theorem 6.3.

Compute and verify solutions

Let $\mathcal{F} \subseteq \mathbb{F}_q[x_{1,1}, x_{1,2}, y_{1,1}, y_{1,2}]$ or $\mathcal{F} \subseteq \mathbb{F}_q[x_2, y_2]$ be the model that we obtained in the previous step for secret P and secret Q, respectively. We enumerate $V(\mathcal{F})(\mathbb{F}_q)$ with appropriate tools.

For secret P, this provides us with an assignment for $(x_{1,1}, x_{1,2}, y_{1,1}, y_{1,2})$ that corresponds to P'. Based on $\psi \circ \phi$ we are then able to compute candidates for P. In the case where Q is the secret, we will directly obtain an assignment for (x_2, y_2) that corresponds to Q. Finally, we are able to verify candidate solutions based on the result of a correct pairing computation.

6.3.4. Discussion

We see that our attack exploits two algebraic properties:

1. The fact that the secret is defined over a proper subfield of $\mathbb{F}_{q^k}$ allowed us to transform our model from $\mathbb{F}_{q^k}$ to $\mathbb{F}_q$ without introducing too many variables. In our special case where we attacked a curve with j-invariant 0 that has a twist of degree 6, $\mathbb{G}_2$ has a compact representation. Therefore, we had to introduce only $k/6 = 2$ variables for each coordinate of elements in $\mathbb{G}_2$ to apply Weil restriction of scalars (cf. Remark 5.17).

2. With $(q^k - 1)/\Phi_k(q) = q^8 + q^6 - q^2 - 1$ this factor of the exponent increases the degree of the model only by a factor of $\mathrm{w}_q(q^8 + q^6 - q^2 - 1) = 4$.

We remark that these properties are also exploited to obtain more efficient pairing computations. The first is used to define the Ate pairing (cf. Definition 2.38), to apply denominator elimination (cf. Remark 3.11), to simplify the computation of $\mathrm{mil}_{n,P}$ (cf. Section 3.1.2), or to speed up arithmetic in $\mathcal{G}_1$ and $\mathcal{G}_2$ [GS08]. The second property is used for an efficient implementation of the final exponentiation (cf. Section 3.2.3 and [Sco+09]).

Our approach to attack the final exponentiation can also be transferred to other parameters and its efficiency only depends on the concrete weight of $(q^k - 1)/\Phi_k(q)$. Furthermore, it can be combined with other techniques from literature to attack the Miller algorithm.

We showed that it is sufficient to invert the heavy factor of the final exponent by means of a fault. Therefore, our approach requires only one erroneous execution to invert the final exponentiation. This is more efficient compared to the attack on the final exponentiation that is proposed in [LFG13] (cf. Section 5.4.2) and that requires three executions. We remark that the two attacks are based on different fault models: We assume instruction skip faults, while [LFG13] assumes faults in $\mathbb{F}_q$.

6.3.5. Simulation based evaluation of the efficiency

We implemented the analysis with Sage [Sag14]. We did not perform the attack in practice. Instead, we simulated the attack, i.e., we assumed a perfect oracle for f_1 and f_2 from (6.11). In our concrete case, we attacked the last iteration with $j = 0$ of the loop in Line 10–Line 15 of Algorithm 6.3 of the exponentiation. We repeated the attack for 10 times for both cases, secret $P \in \mathbb{G}_2$ and secret $Q \in \mathbb{G}_1$. In both cases, we select secret and public argument of the pairing uniformly at random in the corresponding groups.

For the simpler case with secret Q where the model contains only two variables, we parameterized the BN curve with $u = 2^{62} - 2^{54} + 2^{44}$ as in [Beu+10]. This results in a 254 bit prime q and an extension field $\mathbb{F}_{q^k}$ of size 3048 bits. For this set of parameters, our assumption $r \nmid \left(\left(q^4 - q^2 + 1\right)/r - 2^j\right)$ is satisfied for iteration $j = 0$. We used Sage's `variety()` function that is based on Groebner bases techniques to compute the secret. For each of the 10 repetitions, we were able to recover the secret Q in a matter of seconds.

For the more complex case with secret P where the model contains four variables, we were not able to perform the analysis with a realistic size of parameters. The reason is that our version (Version 6.1) of Sage offers no efficient implementation to solve non-linear equations with more than two variables for fields of size $q > 2^{30}$. Hence, we chose $u = 78$. This results in a q of size 30 bits. In this case, our assumption $r \nmid \left(\left(q^4 - q^2 + 1\right)/r - 2^j\right)$ is also satisfied for iteration $j = 0$. As in the previous case, the analysis including the computation of the secret takes only seconds. We remark that according to [Ver08], the complexity of solving a system of non-linear equations over finite fields $\mathbb{F}_q$ is dominated by the number of monomials in the system and not by the size of q. Hence, we can expect that our analysis is also efficient in the case of secret P for larger fields $\mathbb{F}_q$, if we use a CAS that supports these fields.

Chapter 7.

Singular curve point decompression fault attacks

In this chapter we present our results from [BG15]. We use instruction skip faults to transfer the DLOG problem from a cryptographically strong elliptic curve to a weak singular curve. We apply our technique to attack the protocols Decompress-And-Multiply and Hash-And-Multiply from Definition 3.16 and Definition 3.18, respectively.

This chapter is structured as follows: In Section 7.1 we introduce invalid point attacks. Then we give an overview of related work and summarize our contribution. We also discuss properties of our attack. In Section 7.2 we introduce our attack for curves with j-invariant 0 (cf. Definition 2.4). In Section 7.3 we provide an attack for general curves in short Weierstrass form. In Section 7.4, we present concrete applications and compare both attacks. In Section 7.5 we discuss the effectiveness of proposed countermeasures against our attacks.

7.1. Introduction

Our attack is an example for an invalid point attack, a special case of a weak curve attack. *Weak curve attacks* are a category of fault attacks on ECC. Here, an attacker tampers with the parameters a_4 or a_6 that define the curve, the coordinates of points, or the curve's field of definition. The effect is that ECSM is performed in a different group where the complexity of solving the DLOG problem is lower. Notable examples for attacks in this category are given in [BMM00], [Ant+03], [CJ05], and [Fou+08a].

Attacks where the weak curve is obtained by modification of the base point of the ECSM are often called *invalid point* attacks [FV12] (see also Definition 3.3). An important observation made in [BMM00] is that many algorithms for ECSM use only the parameter a_4, and not the parameter a_6 of the curve (cf. Remark 2.10 on page 17). This fact can be exploited in an attack: A fault is introduced to move the base point onto a weak curve that shares the parameter a_4 with the original curve but uses a different parameter a_6:

$$\begin{array}{ccccc} E : y^2 = x^3 + a_4x + a_6 & & & & \tilde{E} : y^2 = x^3 + a_4x + \tilde{a}_6 \\ P & & \rightsquigarrow \text{fault} \rightsquigarrow & & \tilde{P} \end{array}$$

Then, an algorithm for computing ECSM on E implicitly performs ECSM of a secret s and $\tilde{P}$ on the weak curve $\tilde{E}$. If the attacker is able to access the result $\tilde{Q} = s\tilde{P}$ he obtains the DLOG instance $(\tilde{P}, \tilde{Q})$ on the weak curve. This result is used to obtain the secret s on the original curve as well. For most previous fault attacks, the notion of a weak curve is that of a curve with smooth composite order, i.e., a curve where the largest prime factor of the curve is upper-bounded by some smoothness parameter. On such a curve, the DLOG can be computed using the Pohlig-Hellman approach.

Based on our outline of invalid point attacks we characterize the party $\mathcal{B}$ of the protocols Decompress-And-Multiply and Hash-And-Multiply:

Definition 7.1. *Consider the party $\mathcal{B}$ of Definition 3.16 or Definition 3.18. If $\mathcal{B}$'s implementation of ECSM does not use the parameter a_6, we call $\mathcal{B}$ invalid point vulnerable (IPV). If $\mathcal{B}$ is IPV and does not implement countermeasures such as point validity checks or randomization, we call $\mathcal{B}$ first order IPV (IPV1). If $\mathcal{B}$ is IPV and outputs Q only after a positive validity check or after de-randomization, we call $\mathcal{B}$ second order IPV (IPV2).*

Note that implementations following standards like IEEE 1363 do not use the parameter a_6 for ECSM (cf. Annex A of [IEE00]) and are IPV. With respect to the validity check, note that Algorithm 3.1 for decompression and Algorithm 3.3 for hashing to an elliptic curve already output points on the correct curve. Furthermore, IEEE 1363 defines validity checks as optional. Hence, such implementations might even be IPV1.

7.1.1. Related work

Various weak curve attacks and also invalid point attacks have been described in the literature. For an overview, we refer to [FV12; Bar+12b; ADH12]. Here, we review the attacks that are most relevant for us. The first invalid point attack that is based on faults was described in [BMM00]. Here, Biehl, Meyer, and Müller show how faults introduced in the coordinates of P prior to the ECSM can be used to mount an invalid point attack. Antipa et al. [Ant+03] present attacks based on weak domain parameters, and in [CJ05] Ciet and Joye showed that faults in any of the domain parameters may result in a weak curve.

Several countermeasures have been proposed against fault attacks on ECSM [FV12; Bar+12b]. The standard approach to defeat invalid point attacks is to check if the result of the ECSM is on the original elliptic curve; if this is not the case, the output is discarded. Furthermore, point compression presented in Section 3.2.1 has sometimes been proposed as a natural countermeasure against invalid curve attacks because decompression assures that the resulting point is on the curve defined by a_4 and a_6.

It was already observed, for example in [Yen+03], that a validity check can be attacked with a second fault. An invalid point attack that shows how validity checks can be circumvented is the *twist curve* attack from Fouque et al. [Fou+08a].

This attack is based on special implementations of ECSM that do not use the y-coordinate of the base point P. The authors use faults to perform the ECSM on a weak twist (cf. Definition 2.13) of the original curve that has smooth order. They furthermore show how a second fault can be used to map the point back to the original curve in order to pass the validity check. This attack has two limitations: First, many standard implementations do not use the required special form of the ECSM. Second, the efficiency of the attack depends on the smoothness of the order of the twist.

To deal with attacks on the validity check, Dominguez-Oviedo and Hasan [DH11] propose a countermeasure that randomizes the base point by adding a random point. In a correct ECSM, the randomness will be removed afterwards by subtracting a matching multiple of the random point. If a fault modifies the base point into a point on a weak curve, the randomness will not cancel out and it is not possible to compute the secret from the erroneous result anymore.

An attack that is related to our attack in the sense that it also reduces the DLOG problem from an elliptic curve to the DLOG problem in a finite field is the famous work of Menezes, Okamoto, and Vanstone [MOV93], called the MOV attack. Here, a bilinear pairing is used to reduce the DLOG problem on $E(\mathbb{F}_q)$ to the DLOG problem in $\mathbb{F}^*_{q^k}$, where k is the embedding degree of q and r (cf. Definition 2.16). This attack was the first application of pairings in cryptography. Different from our work, the MOV attack does not use faults and singular curves and is efficient only if k and hence $\mathbb{F}_{q^k}$ is small. As a consequence of the attack, supersingular curves from Definition 2.8 are now avoided for standard ECC because they have bounded embedding degree (cf. Theorem 3.5).

In the weak curve attacks that we described so far, the notion of a weak curve is a curve with smooth order. As we explain in the next section, singular curves (cf. Section 2.2.3) are also weak curves and singular curves were already proposed before our work for an invalid point attack by Karabina and Ustaoglu in [KU10]. In [KU10] it is assumed that the attacker is able to directly choose the base point of the ECSM on the singular curve. For implementations with point validity checks or point compression, this is not possible and the attack often does not apply in practice. Furthermore, and different from our work, Karabina and Ustaoglu do not consider fault attacks to circumvent this limitation.

7.1.2. Our contribution

Let $E/\mathbb{F}_q$ be an elliptic curve in short Weierstrass form (2.3). Let the characteristic of $\mathbb{F}_q$ be $p > 3$. Assume a point $P \in E(\mathbb{F}_q)$ is decompressed with Algorithm 3.1 the major building block of point encoding (cf. Section 3.2.1). Our main contribution is to show how instruction skip faults at Algorithm 3.1 can be used to decompress P to an invalid non-singular point $\tilde{P}$ on a singular curve $\tilde{E}(\mathbb{F}_q)$.

For our attack, we distinguish the two cases $a_4 = 0$ and $a_4 \neq 0$ with a_4 as in (2.3) on page 16. According to Definition 2.4, for $p > 3$ it holds that $a_4 = 0$ if and only if the j-invariant of E equals $j = 0$. Based on Theorem 2.11, the DLOG problem

on a singular curve defined over $\mathbb{F}_q$ can be transferred to the DLOG problem in $\mathbb{F}_q^+$, $\mathbb{F}_q^*$, or $\mathbb{F}_{q^2}^*$, respectively. The case $\mathbb{F}_q^+$ occurs for curves with $j = 0$ and the cases $\mathbb{F}_q^*$, or $\mathbb{F}_{q^2}^*$ occur for $j \neq 0$. For $\mathbb{F}_q^+$, the DLOG can be computed with a simple division in $\mathbb{F}_q$ and hence can be performed in time $O(\log(q)^3)$. Let

$$L_N(a, c) = \exp(c \log(N)^a (\log \log N)^{1-a})$$

be the sub-exponential function (cf. Definition 15.1.5 of [Gal12]). For computing the DLOG in $\mathbb{F}_q^*$ and $\mathbb{F}_{q^2}^*$ there exists a constant c and sub-exponential time algorithms with heuristic complexity in $L_q(1/3, c + o(1))$ and $L_{q^2}(1/3, c + o(1))$, respectively [Jou+06]. In all three cases $\mathbb{F}_q^+$, $\mathbb{F}_q^*$, or $\mathbb{F}_{q^2}^*$, the DLOG can be computed much faster than the DLOG on an elliptic curve $E(\mathbb{F}_q)$. Hence, a singular curve is an example for a weak curve.

We apply our technique to attack the protocols Decompress-And-Multiply and Hash-And-Multiply from Definition 3.16 and Definition 3.18, respectively. These protocols compute $Q = sP$ for $P, Q \in E(\mathbb{F}_q)$ and for a secret $s \in \mathbb{Z}$. Both protocols have applications in ECC and PBC and hence, we can turn our attack into an attack on real schemes like the BLS short signature scheme from Definition 3.15. In this chapter, we concentrate on the theoretical background of the attack. In Section 8.6 we present our practical realization of the attack for this particular scheme.

7.1.3. Properties of our attack

Note that in an attack on Decompress-And-Multiply, $\mathcal{A}$ in the role of an attacker can choose the input of decompression at $\mathcal{B}$. This gives $\mathcal{A}$ strong control over the input of Algorithm 3.1. In an attack on Hash-And-Multiply, $\mathcal{A}$ has only weak control over the input of Algorithm 3.1. Here, Algorithm 3.1 is used as a building block of Algorithm 3.3 and the input of Algorithm 3.1 is hashed before decompression.

We show that for every elliptic curve, there exists a set of inputs for point decompression that is vulnerable to instruction skip faults. That means that for every element of this set, we identify an instruction in the decompression algorithm such that if this instruction is skipped, decompression of the corresponding element yields a point on a singular curve. For arbitrary elliptic curves in short Weierstrass form (2.3), the set of vulnerable elements is relatively small. Hence, to mount our attack, strong control is required for explicitly selecting one of those elements and the attack applies only to Decompress-And-Multiply. For curves with j-invariant 0, it turns out that half of the inputs of point decompression are vulnerable. Therefore, it is possible for an attacker with weak control to choose input strings that are hashed to vulnerable elements. Hence, for those curves, the attack also applies to Hash-And-Multiply.

With respect to efficiency, our attack is more efficient for curves with j-invariant 0 than for general curves. This is because for general curves, our attack results in a singular curve that is isomorphic to a multiplicative subgroup of a finite field,

where for curves with j-invariant 0, the resulting curve is isomorphic to the additive group of a finite field where the DLOG problem is trivial. In summary, for curves with j-invariant 0, our attack has weaker assumptions on the protocol and is more efficient compared to general curves.

We remark that curves with j-invariant 0 are popular in ECC and especially PBC because they have useful properties. First, they admit non-trivial endomorphisms that can be used to speed up the ECSM [GLV01; GS08]. Secondly, they are the only curves with twists of degree 6 (cf. Theorem 2.14). Therefore they yield the most efficient pairings [CLN10] (see also Remark 2.22 on page 20 and Section 3.1.2). Finally, supersingular curves, which are required to define Type 1 pairings from Definition 3.1, often have j-invariant 0 (cf. Theorem 3.4). Consequently, curves with j-invariant 0 are defined in standards like [Cer10]; used in applications like Bitcoin [Bit15]; considerable effort has been spent to construct them for application in PBC [BN06; KSS08]; and they are proposed in, e.g., [BF03] and RFC-5091 [BM07].

The efficiency of our attack for curves with j-invariant 0 also has practical advantages. In many scenarios, first the DLOG has to be computed on the weak curve to obtain a candidate for the secret. Then the candidate has to be verified, e.g., based on the public key. If the fault mechanism is not very precise, a large number of experiments has to be performed before a point on the weak curve is obtained. Consequently, a large number of candidate DLOG instances have to be solved. This is easily possible for our attack on curves with j-invariant 0, where the DLOG problem on the resulting singular curve is trivial. For our attack on general curves or for attacks that rely on invalid points with smooth order like [BMM00], their sub-exponential run time may not be efficient enough in practice.

Usually, in our attack we are able to compute the DLOG from a single faulty run of the protocol on a vulnerable element. This makes our attack applicable to protocols where the exponent is a nonce and that allow us to recover the secret key from a complete nonce. To recover the complete nonce from a single run, the attack in [BMM00] requires that the invalid point has a large but smooth order. The probability to obtain such a point is relatively small. Consequently, the expected number of experiments to recover the secret is larger than in our case, and the attack is possibly harder to realize in practice.

Furthermore, the invalid point attack from [BMM00] applied to curves with j-invariant 0 may fail. If the orignal curve has j-invariant 0, the invalid points in [BMM00] are on an elliptic curve that also has j-invariant 0. This implies that the faulty curve is a twist of the original curve (see Section 2.2.4). If the twist does not have a smooth order, the attack is not successful. In [Fou+08a] the invalid points are always on the twist of the original curve, independent of the j-invariant. As for [BMM00], the attack fails if the twist does not have a smooth order. Our attack does not require curves with weak twists and is even more efficient for curves with j-invariant 0.

Because point decompression is also used in Algorithm 3.2 for random point sampling, our attack also applies to random point sampling for curves with j-

invariant 0. This allows us to defeat the randomization countermeasure from [DH11]. The basic idea is to attack decompression with one fault and random point sampling with another fault to obtain a randomized point on the singular curve.

7.2. Attacks for curves with j-invariant 0

Now we present our attacks for curves with j-invariant $j = 0$, i.e., $E : y^2 = x^3 + a_6$. The attack can be applied to Decompress-And-Multiply (cf. Definition 3.16) and Hash-And-Multiply (cf. Definition 3.18). We only present the attack on Hash-And-Multiply in detail, because the attack on Decompress-And-Multiply follows trivially from our description (see Remark 7.4 at the end of this section).

For the attack, assume domain parameters $D = (\mathbb{F}_q, E, R, r, c)$ (cf. Definition 3.2) where $\mathbb{F}_q$ is of characteristic $p > 3$. We assume that $E : y^2 = x^3 + a_6$ is defined over $\mathbb{F}_p$, i.e., $a_6 \in \mathbb{F}_p$ with $\gcd(p, cr) = 1$ and that $r | \#E(\mathbb{F}_p)$. Note that the case $p \neq q$ occurs in PBC where $q = p^k$ and k is the embedding degree of p and r.

7.2.1. Description of the attack

In the attack, $\mathcal{A}$ takes the role of the attacker and performs the following steps in the protocol Hash-And-Multiply with $\mathcal{B}$:

A-1 For given $t \in \mathbb{N}$, select messages $M \in \{0,1\}^*$ until a message with

$$\mathrm{H}\left(M.\lceil 0 \rceil_2^t\right) = (x_U, b) \text{ and } \sqrt{x_U} \in \mathbb{F}_q^*$$

has been found.

A-2 Send M to $\mathcal{B}$. While $\mathcal{B}$ computes HashToCurve(M), mount an instruction skip fault at Line 5 of Algorithm 3.1.

A-3 For $\mathcal{B}$'s output $\tilde{Q} = (x_Q, y_Q)$ return key candidates s_1 and s_2 with

$$s_1 = (-1)^b \sqrt{x_U} x_Q (y_Q c)^{-1} \mod p, \qquad s_2 = s_1 + p.$$

7.2.2. Properties of the attack

The following theorem summarizes the effectivity of the attack:

Theorem 7.2. *If $\mathcal{B}$ is IPV1 (see Definition 7.1) with secret s and if $\mathcal{A}$'s instruction skip in step A-2 successfully skips the addition $v + a_6$ in Line 5 of Algorithm 3.1, then either $s_1 = s \mod r$ or $s_2 = s \mod r$. Furthermore, all computations of $\mathcal{A}$ can be performed in time $O((\log q)^3)$.*

Proof. First note that in step A-1, $\mathcal{A}$ chooses $\mathcal{B}$'s input M such that $(x_U, b) = \mathrm{H}\left(M.\lceil 0 \rceil_2^t\right)$ with a square x_U. According to Definition 3.18, $\mathcal{B}$ will now execute Algorithm 3.1 as a subroutine of HashToCurve to compute Decompress(x_U, b). Assume $\mathcal{A}$'s instruction skip in step A-2 is successful and $v + a_6$ in Line 5 of

Algorithm 3.1 is skipped. Because $a_4 = 0$, in this case, the value of v in Line 6 of Algorithm 3.1 will be $v = x_U^3$. Because x_U is a square in $\mathbb{F}_q^*$, v is also a square in $\mathbb{F}_q^*$. Hence, the output of the erroneous execution of Algorithm 3.1 will be $\tilde{U} = (x_U, (-1)^b x_U \sqrt{x_U})$. This point is an element of the singular curve $\tilde{E} : y^2 = x^3$. Furthermore, with $x_U \in \mathbb{F}_q^*$ it holds that $x_U \neq 0$ and hence $\tilde{U}$ is not the singular point $(0, 0)$, i.e., $\tilde{U} \notin E_{ns}$.

If $\mathcal{B}$ is IPV, the algorithm for ECSM used by $\mathcal{B}$ does not depend on a_6. Hence, $\mathcal{B}$ will perform ECSM with the cofactor c in Line 6 of Algorithm 3.3 and ECSM with s in step 3 of Definition 3.18 in E_{ns}. With $\tilde{P} = c\tilde{U} \in E_{ns}$ it follows that $\mathcal{B}$ will compute $\tilde{Q} = s\tilde{P} \in E_{ns}$. Because $\mathcal{B}$ is assumed to be IPV1, $\mathcal{B}$ will return $\tilde{Q}$, even if it is not on E.

Now we use the group isomorphism $\phi^+ : E_{ns}(\mathbb{F}_q) \to \mathbb{F}_q^+$, $(x, y) \mapsto x/y$ from Theorem 2.11 to map the DLOG instance to $\mathbb{F}_q^+$. We obtain the two equations

$$\phi^+(\tilde{Q}) = x_Q/y_Q$$
$$\phi^+(\tilde{Q}) = s\phi^+(\tilde{P}) = sc\phi^+(U) = sc(-1)^b/\sqrt{x_U}.$$

If we solve them for s we obtain $s = (-1)^b \sqrt{x_U} x_Q (y_Q c)^{-1} \mod d$, where d is the order of $\phi^+(\tilde{P}) = c(-1)^b/\sqrt{x_U}$ in $\mathbb{F}_q^+$. With $\gcd(p, cr) = 1$ it holds that $d = p$. Because we assumed that $r|\#E(\mathbb{F}_p)$ the Hasse bound from Theorem 2.6 implies $r < 2p$. Therefore, either $s_1 = s \mod r$ or $s_2 = s \mod r$.

With respect to the time complexity, note that multiplication and inversion in $\mathbb{F}_q$ can be performed in $O((\log q)^3)$. □

Note that it is easy for $\mathcal{A}$ to choose a message M in step A-1 such that it results in a square x_U. This is because in $\mathbb{F}_q$ every second element is a square. Hence, we can reasonably expect that a hash function H approximately maps every second message to a square x_U.

We see from the proof of Theorem 7.2 that we recover s only modulo p. Hence, it is crucial for the attack to work over fields of large characteristic p with $r|\#E(\mathbb{F}_p)$. But if this is the case, one run of the protocol is enough to recover $s \in \mathbb{Z}/r\mathbb{Z}$ up to one bit that selects between s_1 and s_2. Hence, the attack is also applicable for protocols where s is a nonce or an ephemeral key that is refreshed in each run of the protocol.

Remark 7.3. The same attack applies if $\mathcal{B}$ uses point compression for the output Q. The reason is that point compression in (3.4) on page 31, different from decompression, does not use the parameters a_4 and a_6 of the original curve E. Hence, in a successful attack, $\mathcal{B}$ will output the compression of $\tilde{Q} \in \tilde{E}$. Then, $\tilde{Q}$ can be recovered in step A-3 of the attack based on the equation of $\tilde{E}$.

Remark 7.4. The attack directly applies to the protocol Decompress-And-Multiply because in this protocol, $\mathcal{A}$ has even strong control over the input of Decompress and is able to directly choose $(x_U, b) \in \mathbb{F}_q \times \{0, 1\}$ with a square x_U as $\mathcal{B}$'s input.

In Section 8.6, we practically perform the attack of this section on the BLS signature scheme from Definition 3.15. In Appendix A.1 we give a numerical example from the practical realization of our attack.

7.3. Attack for general curves

We now present an attack on Decompress-And-Multiply that applies to general curves in short Weierstrass form $E : y^2 = x^3 + a_4x + a_6$ with j-invariant not necessarily 0. Based on the ideas of the previous section our aim is to introduce an error such that the ECSM is performed on the singular curve $\tilde{E} : y^2 = x^3 + a_4x + \tilde{a}_6$ with discriminant $\Delta = -16(4a_4^3 + 27\tilde{a}_6^2) = 0$ (cf. Definition 2.4).

For efficiency reasons, nearly all standardized curves with $a_4 \neq 0$ have $a_4 = -3$. To make our description more concrete, we focus on these curves throughout this section but our attack can also be generalized to other curves. It follows from $\Delta = 0$ that the corresponding singular curves are given as $\tilde{E} : y^2 = x^3 - 3x \pm 2$.

7.3.1. Description of the attack

To describe our attack, we define the set $\mathcal{F} \subset \mathbb{F}_q$ of faults that we can introduce at the computation $v \leftarrow v + a_4$ in Line 3 of Algorithm 3.1. We add δ to $\mathcal{F}$ if we are able to perform an instruction skip fault that modifies $v \leftarrow v + a_4$ into $v \leftarrow v + a_4 + \delta$. For example, in the case $a_4 = -3$ the OpenSSL implementation[1] computes $x^3 - 3x + a_6$ as $x^3 - (2x + x) + a_6$. Now assume we are able to skip the addition of x or the addition of $-(2x + x) = -3x$. Then either $x^3 - 2x + a_6$ or $x^3 + a_6$ is computed and we obtain $\mathcal{F} = \{1, 3\}$.

In the attack, $\mathcal{A}$ takes the role of the attacker and performs the following steps in the protocol Decompress-And-Multiply with $\mathcal{B}$:

A-1 If $\mathcal{F}$ is empty fail.

A-2 Remove an element δ from $\mathcal{F}$ and define $x_i = ((-1)^i 2 - a_6)/\delta$ for $i = 0, 1$.

A-3 Select $i \in \{0, 1\}$ such that $x_i^3 + (a_4 + \delta)x_i + a_6$ is a square in $\mathbb{F}_q^*$. If no such i exists go back to step A-1.

A-4 Set $b = 0$ and send (x_i, b) as input to $\mathcal{B}$.

A-5 While $\mathcal{B}$ decompresses (x_i, b), mount an instruction skip fault at Line 3 of Algorithm 3.1 in order to replace the addition with a_4 by an addition with $a_4 + \delta$.

A-6 Define $y_i = \sqrt{x_i^3 - 3x_i + (-1)^i 2}$, $\tilde{P} = (x_i, y_i)$, $x_S = (-1)^i$, and $\alpha = \sqrt{3x_S}$. With ϕ^* from Theorem 2.11 output the DLOG s' of $\phi^*(\tilde{Q})$ to the basis $\phi^*(\tilde{P})$ in $\mathbb{F}_q(\alpha)^*$ as a key candidate.

[1] we checked Version 1.0.2

7.3.2. Properties of the attack

The following theorem summarizes the effectivity of the attack:

Theorem 7.5. *If $\mathcal{B}$ is IPV1 (see Definition 7.1) with secret s and if $\mathcal{A}$'s instruction skip in step A-5 successfully replaces the addition $v + a_4$ with $a_4 = -3$ in Line 3 of Algorithm 3.1 by $v + a_4 + \delta$, then $\mathcal{A}'s$ output fulfills $s' = s \mod d$ where d is the order of $\phi^*(\tilde{P})$ in the multiplicative group $\mathbb{F}_q(\alpha)^*$.*

Proof. If the instruction skip in Line 3 of Algorithm 3.1 is successful, and the addition $v + a_4$ in Line 3 of Algorithm 3.1 is replaced by the addition $v + a_4 + \delta$, the value of v in Line 6 of Decompress will be $x_i(x_i^2 + a_4 + \delta) + a_6$. Then, the check in step A-3 guarantees that this is a square and hence, the output of Decompress will be (x_i, u) with $u = (-1)^b\sqrt{x_i(x_i^2 + a_4 + \delta) + a_6}$.

We will now show that $u = y_i$ and hence that Algorithm 3.1 will output $\tilde{P}$. With $\mathcal{A}$'s choice $x_i = ((-1)^i 2 - a_6)/\delta$ and with $a_4 = -3$, we obtain

$$u^2 - y_i^2 = \delta x_i + a_6 - (-1)^i 2 = \delta\frac{(-1)^i 2 - a_6}{\delta} + a_6 - (-1)^i 2 = 0.$$

With $b = 0$ it follows that $u = y_i$ and hence Algorithm 3.1 outputs $\tilde{P}$.

From the definition of y_i, we see that this point is on the curve $\tilde{E} : y^2 = x^3 - 3x + (-1)^i 2$. The discriminant of $\tilde{E}$ is 0 and it follows that $\tilde{E}$ is singular. Because step A-3 ensures $y_i \neq 0$ it follows from Theorem 2.11 that $\tilde{P}$ is non-singular and hence $\tilde{P} \in E_{ns}(\mathbb{F}_q)$. Furthermore, if $\mathcal{B}$ is IPV1, it follows that $\mathcal{B}$ will compute and output $\tilde{Q} = s\tilde{P} \in \tilde{E}_{ns}$.

Finally, in step A-6, $\mathcal{A}$ will compute the DLOG in the subgroup of $\mathbb{F}_q(\alpha)^* \subseteq \mathbb{F}_{q^2}^*$ that is generated by $\phi^*(\tilde{P})$. With Theorem 2.11 and because the order of $\tilde{P}$ is d, $\mathcal{A}$'s output s' will satisfy $s' = s \mod d$. □

We see that Theorem 7.5 is not as explicit as Theorem 7.2 in two aspects. First, it does not guarantee that the order of $\phi^*(\tilde{P})$ is large as it has been for the order of $\phi^+(\tilde{P})$ in Section 7.2. It follows that we can not guarantee that enough bits of s are recovered by the attack. Secondly, the sub-exponential complexity for computing the DLOG in $\mathbb{F}_q(\alpha)^*$ can only be shown based on heuristic assumptions [Jou+06]. Nevertheless, for our practical attacks we can assume that $d \approx p$ and that the complexity of DLOG in $\mathbb{F}_q^*$ and $\mathbb{F}_{q^2}^*$ is in $L_q(1/3, c + o(1))$ and $L_{q^2}(1/3, c + o(1))$, respectively.

Remark 7.6. The attack also applies to protocols where $\mathcal{A}$ is able to provide an uncompressed input P to $\mathcal{B}$ and instead, $\mathcal{B}$ aborts if P is not on E. In this case, $\mathcal{B}$ will validate that $y^2 = x^3 + a_4x + a_6$ holds for the input P from $\mathcal{A}$. In an attack, $\mathcal{A}$ provides the input $\tilde{P} = (x_i, y_i)$ from step A-6 as $\mathcal{B}$'s input. Then $\mathcal{A}$ attacks the addition with a_4x_i in the check $y_i^2 = x_i^3 + a_4x_i + a_6$ such that $\mathcal{B}$ checks if $y_i^2 = x_i^3 + (a_4 + \delta)x_i + a_6$ holds. From the proof of Theorem 7.5 it follows that the check will pass for the point $\tilde{P}$.

Note that Remark 7.3 from Section 7.2 for the case where $\mathcal{B}$ outputs Q in compressed form also applies for the attack in this section.

To demonstrate that the attack is efficient for parameters of practical relevance, we performed the analysis for the curve `secp192r1` from [Cer10] that is defined over a 192 bit prime field $\mathbb{F}_q$. We were able to compute the DLOG in $\mathbb{F}_q^*$ with the function `znlog` of Pari/GP in 39 hours on one core of a 64 bit Intel Core i5 CPU with 8 GB of main memory. We give the complete numerical example in Appendix A.2

7.4. Concrete applications of our attacks

In this section, we point to concrete applications of our attacks on the abstract protocols Decompress-And-Multiply and Hash-And-Multiply. Furthermore, we compare the attack from Section 7.2 with the attack from Section 7.3 and summarize their most important properties.

7.4.1. Pairing-based signature schemes

In Section 3.3.2 we introduced pairing-based signature schemes of the type Hash-And-Multiply that hash the signed message to a point on the curve as input for ECSM. Examples for such schemes that are susceptible to our attack from Section 7.2 are given in [BLS04b; Bol03; LQ04]. Also, it was explicitly proposed to instantiate such schemes with curves of j-invariant 0 [Cha+10]. The popular combination of pairing-based signature schemes with curves of j-invariant 0 is an important application of our attack. This also motivates our practical attack on an instantiation of the pairing-based BLS short signature scheme from Definition 3.15 with BN curves from Theorem 3.6 that have j-invariant 0.

The identity-based signature scheme from [CC03] that we also outlined in Section 3.3.2 has a different structure. Here, not the message but the identity of the signer $\mathcal{B}$ is hashed: $\mathcal{B}$ with identity `ID` first computes $P = \text{HashToCurve}(\texttt{ID})$, then computes $Q = sP$ for a nonce s, and finally returns Q as part of the signature. In an attack on ECSM, $\mathcal{A}$ recovers the nonce s first. From s and the corresponding signature, the secret key $D_{\texttt{ID}}$ of $\mathcal{B}$ can be computed. Our attack requires only one invocation of the protocol. Hence, on the one hand, our attack is possible even though ECSM is performed with a nonce. On the other hand, the attack can be applied only if $P = \text{HashToCurve}(\texttt{ID})$ was not precomputed offline by $\mathcal{B}$.

7.4.2. Encryption schemes with point compression

In elliptic curve based instantiations of encryption schemes like Elgamal encryption, typically $Q = sP$ is computed as part of the decryption. Here, $s \in \mathbb{Z}$ is the recipients secret key, $P \in E$ is a component of the ciphertext, and Q is an ephemeral key that is used to recover the message. Furthermore, implementations often use point compression to represent the component P of the ciphertext. Therefore, these

implementations are possible candidates for our attacks on Decompress-And-Multiply.

In the abstract protocol Decompress-And-Multiply $\mathcal{A}$ has access to the result Q of ECSM. Furthermore, the knowledge of Q is required to mount our proposed attacks. But in real-world applications like Elgamal, decryption queries do not provide these elements directly. Instead, a hash function or a KDF is applied to the output Q of ECSM and this value is released. Hence, schemes like Elgamal are vulnerable to our attack, but only if the attacker is able to invert the KDF or is able to access the result of the ECSM prior to the KDF. Nevertheless, our attacks show that the result of ECSM with the secret key has to be protected in an implementation because it is a valuable target for FAs. This situation is similar to the situation for active attacks on Pair-Argument as discussed in Section 5.1.5.

7.4.3. Signature schemes with point compression

A direct application of our attack on Decompress-And-Multiply is the blind signature scheme from Section 5 of [Bol03]. Here, P is the representation of a blinded message and the blind signature under P and secret key s is $Q = sP$. If $\mathcal{A}$ asks $\mathcal{B}$ for the generation of a blind signature under P, and if P is compressed, then we can apply our attacks from Section 7.2 and Section 7.3.

Our next example is ECDSA [ANS99]. Let $D = (\mathbb{F}_q, E, R, r, c)$ be ECC domain parameters, H a cryptographic hash function, and $d \in \mathbb{Z}/r\mathbb{Z}$ be the secret key of $\mathcal{B}$. To compute a signature for message M under d, $\mathcal{B}$ first chooses a nonce k uniformly at random from $[0, r-1]$. Then $\mathcal{B}$ computes $Q = kR$ and returns $\sigma = (x(Q) \bmod r, k^{-1}(\mathrm{H}(M) + dx(Q)) \bmod r)$ as the signature. Because $r \approx q$, $x(Q)$ and hence Q can be recovered from $x(Q) \bmod r$. If R is stored in compressed form $(x_R, b) = \mathrm{Compress}(R)$, we can consider ECDSA signature generation as a restricted case of Decompress-And-Multiply by setting $P = R$ and $s = k$. Here, the restriction is that the point R is fixed as part of the ECC domain parameters and hence, an attacker has *no* control over the input of Algorithm 3.1. We now argue that this is sometimes sufficient to apply our attack.

In the attack from Section 7.2 the adversary uses his weak control over the input of Algorithm 3.1 to choose a square x_U. Hence, if x_R happens to be a square and if the curve has j-invariant $j = 0$ we can still attack the decompression of (x_R, b) as in Section 7.2.1. For example, if ECDSA is instantiated with the curve `secp256k1` from [Cer10] it is potentially vulnerable to the attack:

1. The curve `secp256k1` has j-invariant $j = 0$ as required for our attack.

2. The x-coordinate x_R of the generator R as specified in [Cer10] is a square in $\mathbb{F}_q$ and hence $(x_R, b) = \mathrm{Compress}(R)$ results in a square x_R.

3. Our attack can recover the scalar in one shot and hence it can be applied to ECDSA where ECSM is performed with a nonce k.

Curve	$E : y^2 = x^3 + a_6$	$E : y^2 = x^3 + a_4x + a_6$
j-invariant	$j = 0$	$j \neq 0$
Section	7.2	7.3
Decompress-And-Multiply	yes	yes
Hash-And-Multiply	yes	no
Random point sampling	yes	no
DLOG problem reduced to	$\mathbb{F}_q^+$	$\mathbb{F}_q^*$ or $\mathbb{F}_{q^2}^*$
Complexity	$\log(q)^3$	$L_q(1/3, c)$ or $L_{q^2}(1/3, c)$

Table 7.1.: Comparison of our singular curve attacks for curves over $\mathbb{F}_q$.

An example that uses the combination of ECDSA and secp256k1 is Bitcoin [Bit15]. But we also remark that we are not aware of any ECDSA implementation that stores the fixed generator R in compressed form.

7.4.4. Random point sampling

The attack from Section 7.2 also applies to random point sampling if implemented similarly to Algorithm 3.2. Instead of hashing the message M to (x_U, b), here $(x_U, b) \in \mathbb{F}_q \times \{0, 1\}$ is sampled uniformly at random until it is a valid compression. Then (x_U, b) is decompressed to obtain a point on the curve. Hence, we can attack the computation of Decompress(x_U, b) with Algorithm 3.1 as in Section 7.2.1. If x_U is a square, we are successful. On expectation, we have to repeat the fault injection twice, because every second element in $\mathbb{F}_q$ is a square.

7.4.5. Comparison of attacks

Table 7.1 shows a summary of our attacks. We see that the attack for curves with j-invariant 0 from Section 7.2 applies to a broader class of protocols compared to the attack for general curves from Section 7.3. For curves with $j = 0$, our attack can be used to attack the protocols Decompress-And-Multiply and Hash-And-Multiply defined in Definition 3.16 and Definition 3.18, respectively. Furthermore, it can also be used to attack Algorithm 3.2 for random point sampling. For general curves the attack only applies to the protocol Decompress-And-Multiply because the attacker needs strong control over the input x of Decompress. If x is the image of a hash function H as in Algorithm 3.3, this control is not provided.

With respect to efficiency, the attack also performs better for curves with $j = 0$ than for general curves. Here, the reason is that in the former case, the DLOG problem is reduced to $\mathbb{F}_q^+$ and in the latter case, the DLOG problem is reduced to $\mathbb{F}_q^*$ or $\mathbb{F}_{q^2}^*$. In $\mathbb{F}_q^+$, computing the DLOG is a trivial inversion while in $\mathbb{F}_q^*$ or $\mathbb{F}_{q^2}^*$, sub-exponential index calculus algorithms like the number field sieve are required. Especially if $\alpha = \sqrt{3x_S}$ from Theorem 2.11 is not in $\mathbb{F}_q$, the DLOG problem is

reduced to $\mathbb{F}_{q^2}^*$. Here, computing the DLOG is cheaper than on the original curve, but depending on the size of q, it still might be too expensive in practice.

7.5. Countermeasures

The first option to defeat fault injection is to detect faults by the hardware. For example, in the case of instruction skips, redundancy can be added to detect the modification of instructions [Bar+06]. Since hardware mechanisms are not always available and also because they sometimes only help against special fault injection techniques, software countermeasures are also required. Several software countermeasures against attacks on ECC-based schemes are known. For a survey, we refer to [FV12]. To demonstrate that countermeasures have to be implemented carefully we discuss two countermeasures that were proposed against invalid point attacks: point validity checks and fault infective computations based on secret sharing. We show that both countermeasures are vulnerable to our attack.

7.5.1. Point validity checks

Especially against invalid point attacks, a point validity check of the result Q was proposed. It basically evaluates the Weierstrass equation (2.3) for the output Q of ECSM to verify that Q is still on E. If the verification fails, the result Q is discarded and a failure message is returned. It was already observed in the context of RSA, that the conditional branch of a decision procedure can be attacked with a second fault [Yen+03; KQ07]. This is exactly what we do in our practical attack in Section 8.6. Our results support the concern of [Yen+03] that it is difficult to implement checking procedures such as point validity checks in a secure way.

Furthermore, it is not enough to protect the check's branch instruction. To give just one example, in our attacks from Section 7.2 and Section 7.3 we could inject the very same fault that we used during decompression of P on the evaluation of (2.3) during the check of Q. Then, Q would pass the test exactly in the case where it is on the singular curve (cf. Remark 7.6) .

7.5.2. Fault infective computations

To deal with attacks on a decision procedure, Yen et al. [Yen+03] propose fault infective computation for Chinese remainder theorem (CRT)-RSA. The idea of fault infective computation is to remove an explicit check. Instead, the computation that involves the secret is performed such that an erroneous result does not leak information about the secret. The idea was adapted to ECSM in [DH11, Section 5] for implementations based on the Montgomery ladder. We abstract from the details of [DH11] that are related to the Montgomery ladder. Then the idea is to use a random point R to split P into two shares $P + R$ and $-R$ prior to the ECSM. Then ECSM is computed on both shares and the results are re-combined: $Q = s(P + R) - sR = sP$. If a fault modifies P into $\tilde{P} \in \tilde{E} \neq E$, $\tilde{P}$ and R are

on different curves. Hence, the addition with R and subtraction with sR are not valid group operations on $\tilde{E}$. Consequently, the mask sR does not cancel out and Q depends on the unknown R.

As noted in Section 7.4.4, sampling $R \in E(\mathbb{F}_q)$ based on Algorithm 3.2 is vulnerable to our attack from Section 7.2 for curves with j-invariant 0. To circumvent the countermeasure in this case, we propose to apply a second order attack. With the first fault, we attack the decompression of P to obtain $\tilde{P}$ on the singular curve $\tilde{E}$. With the second fault, we attack the sampling of R to obtain $\tilde{R} \in \tilde{E}$. If both faults are successful all subsequent group operations are performed on the singular curve. Hence, the output of the masked computation is $\tilde{Q} = s(\tilde{P} + \tilde{R}) - s\tilde{R} = s\tilde{P} \in \tilde{E}$. Then we can apply our reduction to the DLOG problem in $\mathbb{F}_q^+$ as in Section 7.2.

As a conclusion, we see that software countermeasures have to be implemented carefully because they can be manipulated with a second fault. This is especially true for curves with j-invariant 0 because in this case, even countermeasures based on fault infective computation are vulnerable to our attack.

Chapter 8.

Practical realization of our fault attacks

In Section 6.2 we presented an FA on Pair-Argument from Definition 3.17 and in Section 7.2 we presented an FA on Hash-And-Multiply from Definition 3.18. There, we described the theoretical analysis of the attacks that is required to recover the secret key from an erroneous computation. We showed that second order attacks are necessary for relevant attacks to either attack the complete pairing computation or to remove countermeasures. In this chapter, we give detailed background information on our practical realization of these attacks based on our work from [Blö+14] and [BG15]. Especially, we show that it is possible to implement second order attacks in the complex scenario of PBC. To realize our attacks, we use clock glitches to implement instruction skip faults. Hence, in this chapter we move on from the mathematical aspects of our attacks to their engineering aspects.

This chapter is structured as follows. In Section 8.1, we discuss related work and our contribution. Then we give some background information on clock glitching, instruction skip faults, and their applications. In Section 8.2 we describe our setup to realize instruction skip faults. In Section 8.3 we explain how to perform second order faults with our setup. In Section 8.4 we give background information on the device and the software that we target in the attack. In Section 8.5 and Section 8.6 we describe our practical realization of the attacks from Section 6.2 and Section 7.2, respectively.

8.1. Introduction

We distinguish between three different terminologies with respect to fault attacks:

1. The fault *injection technique* that is used to generate a fault like for example clock glitches, voltage glitches, electromagnetic pulses, laser beams, or heating.

2. The *effect* of the fault on the computation and on data. Possible effects include instruction replacement, instruction skip, erroneous load/store of data, and erroneous branching.

3. The *exploitation* of the fault effect to recover the secret. This includes for example, algorithm specific attacks, differential fault analysis, and safe-error attacks.

In this thesis, we use clock glitching as fault injection technique to implement instruction skip faults. Furthermore, we exploit these faults in the algorithm specific attacks from Section 6.2 and Section 7.2 on the pairing computation and on point encoding in combination prior to ECSM.

8.1.1. Related work

There is a large amount of work on fault attacks with contributions to injection techniques, fault effects, and fault exploitation. For an overview, we refer to [Bar+06; Bar+12b; Bar+12a; KSV13].

In the following, we give some exemplary references related to clock glitching and instruction skip faults. Furthermore, we recall the few practical realizations of fault attacks in the context of PBC.

Clock glitching

Clock glitching is a standard technique to overclock an electronic circuit like the CPU of a smart card controller for a short period. This technique has been used as a fault injection technique already for a long time [AK96]. Overclocking causes violations in the setup time of flip-flops. For a brief introduction we refer to [Ago+10]. Clock glitching may have different effects like, e.g.:

1. Erroneous load or store of data from or to memory.
2. Erroneous fetching, decoding, or execution of instructions.
3. Erroneous updates of the program counter.

In this thesis, we target the AVR ATxmega128A1 from Atmel's AVR family [Atm13]. In [BGV11] the effect of clock glitching on the ATmega163 from the Atmel AVR family was studied. The work provides an extensive analysis how different instructions are effected depending on various parameters of the clock glitch. This includes the description of an FPGA setup that is used to generate clock glitches. Also background information on the AVR architecture is provided to explain the observed effects.

Note that access to the clock of the target CPU is required to apply clock glitching. We remark that several smart card controllers do not have an accurate internal oscillator. Hence, they use an external clock source. Furthermore we view a successful attack that is based on clock glitching as an indication for a vulnerability also against other mechanisms like, e.g., voltage glitching.

Instruction skip faults

Concerning the effects of faults, we consider instruction replacement faults, or more specifically, instruction skip faults. *Instruction replacement faults* are faults where a fault is injected during instruction fetch, instruction decode, or instruction

execution with the effect that another instruction is executed in place of the original instruction. An *instruction skip* fault is the special case where the original instruction is not executed at all or replaced with an instruction that does not effect the data of the cryptographic algorithm. On some CPUs, for example from the Atmel AVR family, the realization of instruction skip faults is simplified because these CPUs ignore invalid opcodes and continue with code execution [BGV11].

Instruction replacement faults can be realized based on clock glitching. Indeed, it has been shown that instruction replacement faults and instruction skip faults can be achieved with various injection techniques and on various computer architectures. Choukri and Tunstall [CT05] use voltage glitches to generate instruction skip faults on a PIC16F877 controller. Furthermore, they exploit the instruction skips to reduce the complexity of AES. In [KQ07] Kim and Quisquater use voltage glitches to generate second order instruction skip faults on an AVR CPU. They use this to attack a protected CRT-RSA implementation. As already mentioned, in [BGV11], instruction skip faults were described as one possible effect of clock glitching on AVR CPUs. Furthermore, in [Riv+15], instruction skip faults on ARMv7-M CPUs were introduced by means of EM pulses.

Fault attacks on PBC

In Section 6.1.1 we gave an extensive overview of how fault attacks were exploited in the context of PBC. Most attacks were only described theoretically and not validated in practice. Two exceptions are [BMH13] and [Las+14].

Bae, Moon, and Ha [BMH13] use a laser pulse to implement an instruction skip fault on an AVR ATmega128L. This fault is used to skip a conditional branch. Bae, Moon, and Ha describe how this fault can be exploited for an attack on a pairing computation. In their practical evaluation, they do not attack a complete pairing computation. Instead, they attack a small test program that generates a trigger for the laser and sends a special output in case the fault injection was successful.

In [Las+14] Lashermes et al. practically validate previous theoretical attacks on the Miller algorithm. The practical evaluation is performed on an ARM Cortex M3 processor and EM pulses are used as the fault injection technique. Lashermes et al. implement two effects: errors at loading data, and skipping of branch instructions. Both effects can be exploited to attack the pairing computation (cf. Chapter 5). As we explained in Section 5.1, most pairings require second order attacks: The first fault is required to target the Miller algorithm and the second fault is required to target the final exponentiation. In [Las+14] only first order attacks on the Miller algorithm are considered. The authors attack their own implementation and precede the target instruction with a trigger and `NOP` instructions. According to [Las+14], this is not necessary but it simplifies the attack in two ways: It facilitates synchronization with the EM pulse and it clears the processor's pipeline for a better reproducibility of the fault.

8.1.2. Our contribution

Based on our results from [Blö+14] and [BG15], we present our practical realization of the attacks from Section 6.2 and Section 7.2, respectively. Both attacks are performed on an AVR ATxmega128A1. With respect to our attack on Pair-Argument from [Blö+14] (cf. Section 6.2) our contribution is summarized as follows:

1. We present the first practical realization of a fault attack against a complete pairing computation. Different from [Las+14] that target only the Miller algorithm, we use a second order attack to target Miller algorithm *and* final exponentiation.

2. We attack an independent pairing implementation from [AG] that we did not modify for a simpler realization of our attack. Especially, the assembly on our final target device does not precede the target instructions with code to simplify synchronization like triggers or `NOP` instructions.

3. We analyze the assembly code that was generated with avr-gcc to identify suitable target instructions for our basic attack from [Blö+14] and our extended attack from Section 6.2. We perform the basic attack in practice.

4. As we explain in Section 5.1, second order attacks are an important tool for the implementation of fault attacks against PBC, or more specifically, against the pairing computation as part of Pair-Argument. The complexity of an attack increases with a higher order. Especially without additional triggers, it becomes more difficult to determine the precise timing of the glitches at the target instructions. In Section 8.3 we give a detailed description of our profiling strategy that we use to determine parameters of the individual glitches.

5. In the practical realization of a second order instruction skip attack, many experiments are required until both target instructions are skipped successfully. Furthermore, it is not trivial to identify successful executions of the attack: For every experiment that generates reasonable output, key candidates have to be computed and verified. Therefore, we automated the analysis of Section 6.2 with Sage in order to perform a large number of experiments without manual intervention.

For the realization of our attack on the BLS signature scheme from [BG15] (cf. Section 7.2) we follow the same paradigm as before and target an independent implementation. Therefore, our contribution here is similar:

1. To the best of our knowledge, we present the first practical realization of a fault attack against a complete implementation of a pairing-based signature scheme. Furthermore, we present the first practical realization of a fault attack that exploits singular curves.

2. We attack the implementation of [AG] of the algorithm Sign from Definition 3.15. We did not modify the implementation for a simpler realization of our attack. Especially, the assembly on our final target device does not precede the target instructions with code to ease synchronization like triggers or `NOP` instructions.

3. We analyze the assembly code that was generated with avr-gcc to identify suitable target instructions.

4. We added a simple countermeasure against first order attacks to the implementation of [AG] and hence, we target the implementation with a second order attack. In Section 8.6 we describe how we determine parameters of the individual glitches.

5. As above, we automated the analysis of Section 7.2 with Sage in order to perform a large number of experiments without manual intervention.

While [BGV11] analyze an AVR ATmega163 that is specified for CPU frequencies up to 8 MHz, we perform our experiments on an AVR ATxmega128A1 that is specified for 32 MHz. Hence, overclocking and therefore also clock glitching is potentially more difficult for the AVR ATxmega128A1. Our results show that similar techniques as in [BGV11] can also be used to attack this device.

For both attacks, we use a setup that is similar to the setup from [BGV11]. Our setup was developed by Gomes da Silva [Gom14] and hence, we do not account the setup as a contribution of this thesis. Nevertheless, our results show that the setup is applicable in complex scenarios and can be used to attack real protocols.

8.1.3. Exploiting instruction skip faults in attacks on PBC

Instruction skip faults are an important tool that can be exploited in various ways in FAs on PBC. In the following, we describe some basic techniques that are based on instruction skip faults and that have applications in PBC. We use some of them in our practical attacks in Section 8.5 and Section 8.6.

Change the number of loop iterations: By skipping a conditional jump at the end of a `for` or `while` loop we can leave the loop prematurely. This approach has been used, for example in [CT05], to reduce the number of AES rounds. We use this approach to realize our fault of Section 6.2, where we leave the `for` loop of Algorithm 6.1 after one iteration. We give more details on the concrete realization in Section 8.5. Furthermore, this technique can also be used to leave the loop in Line 10–Line 15 of Algorithm 6.3 to realize our attack on the final exponentiation from Section 6.3.

By skipping updates of loop counters, we can perform an additional iteration of a loop. Based on this, the attack of [PV06] that we described in Section 5.4.1 can be realized.

Skipping conditional branches: By skipping a conditional branch instruction or the update of a `zero` flag, we can also skip `if` or `else` branches. We use this technique in Section 8.6 to eliminate the point validity check in the practical realization of our attack from Section 7.2 (see also Remark 8.3 in Section 8.6.2). Furthermore, this technique can also be used to skip the `if` branch in Line 12 of Algorithm 6.3 for realizing our attack on the final exponentiation from Section 6.3. Another application is the attack from [BMH13] (cf. Section 8.1.1).

Skipping finite field operations: By skipping function calls, we can skip complete sub-routines like finite field operations. For example, in [SH08] RSA signatures are attacked by skipping a squaring operation. To realize our attack on the final exponentiation from Section 6.2, on the Ate pairing from Section 6.3, or on point decompression from Section 7.2, we skip a finite field multiplication or a finite field addition, respectively. We give more details on the realization of skipping instructions calls in Section 8.5 and Section 8.6.

Data faults at finite field elements: It is also possible to realize faults in $\mathbb{F}_q$ based on instruction skips. During an initialization or update of a field element that usually is organized as an array of machine words, we can skip the update of one word. This introduces a localized fault. Often, if a word is of width 16 bit or less, this even allows us to guess the introduced fault. This is advantageous for the analysis of fault attacks on pairings (cf. Section 5.3.7).

Redirecting output pointers: Another effect that we can realize with instruction skip faults is to redirect pointers to arguments of functions. Assume we partially skip the initialization of a pointer. For example, we might zero the least significant word of the pointer. Subsequent updates of the variable that is referenced by this pointer will then modify a different memory location.

For an example, consider the variable b in Algorithm 3.4 that is used to accumulate the result. Assume we modify the pointer that references this variable at the end of the first iteration. Then, the loop is executed as usual, but the variable b is not updated. This has the effect of virtually leaving the loop after one iteration. This is useful to circumvent countermeasures that, for example, test the integrity of the loop counter or the point R of Algorithm 3.4. This technique can also be used to realize our practical attacks as we explain in Remark 8.1 and Remark 8.2 of Section 8.5 and Section 8.6, respectively.

8.2. Experimental setup

In this section, we describe the setup that we use for CPU clock glitching. It was developed as part of a Bachelor's thesis and for more background we refer the reader to [Gom14]. The setup is not specialized to attacks on PBC and can be used in other scenarios. It consists of three main components: the glitcher, the

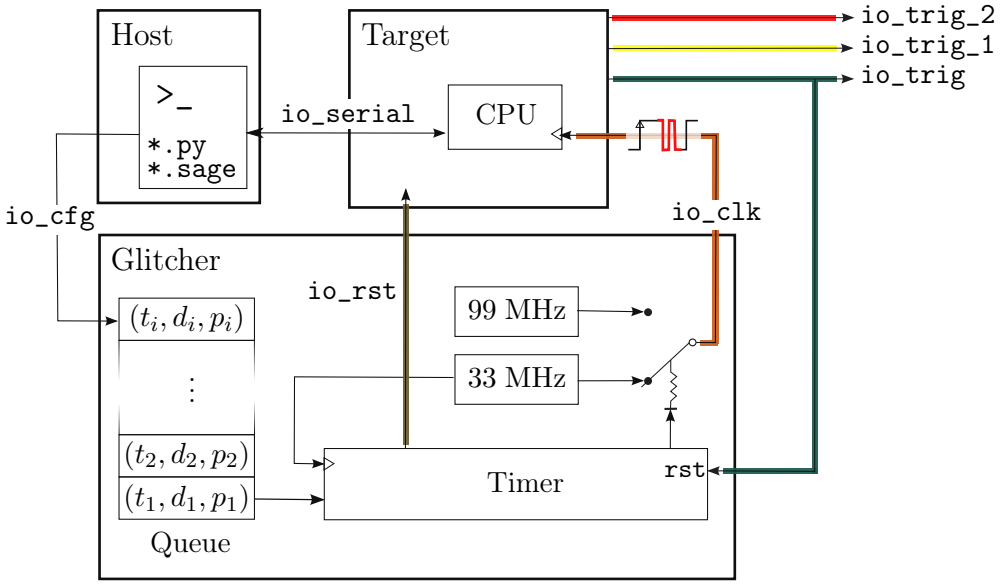

Figure 8.1.: Simplified block diagram of our setup. The host loads the queue of the glitcher via `io_cfg`. The glitcher resets the target via `io_rst` and synchronizes at `io_trig` with the target. According to its configuration queue it generates the glitches on the external clock `io_clk` of the target. The target executes the program under attack and communicates on the serial line `io_serial` with the host. The signals `io_trig_1` and `io_trig_2` are used for profiling and are implemented only by special devices (cf. Section 8.3).

host system, and the target. A block diagram of the setup is shown in Figure 8.1, and Figure 8.2 shows a picture of the setup. 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 the glitcher and to acquire the output of the device under attack. The target executes the attacked program. We now describe the three components individually.

8.2.1. Clock glitcher

For the hardware of the glitcher we use the DDK [NS13]. This is a low-cost, open source development platform which consists of an FPGA and an ARM CPU. 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 allows to easily automate the setup.

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.

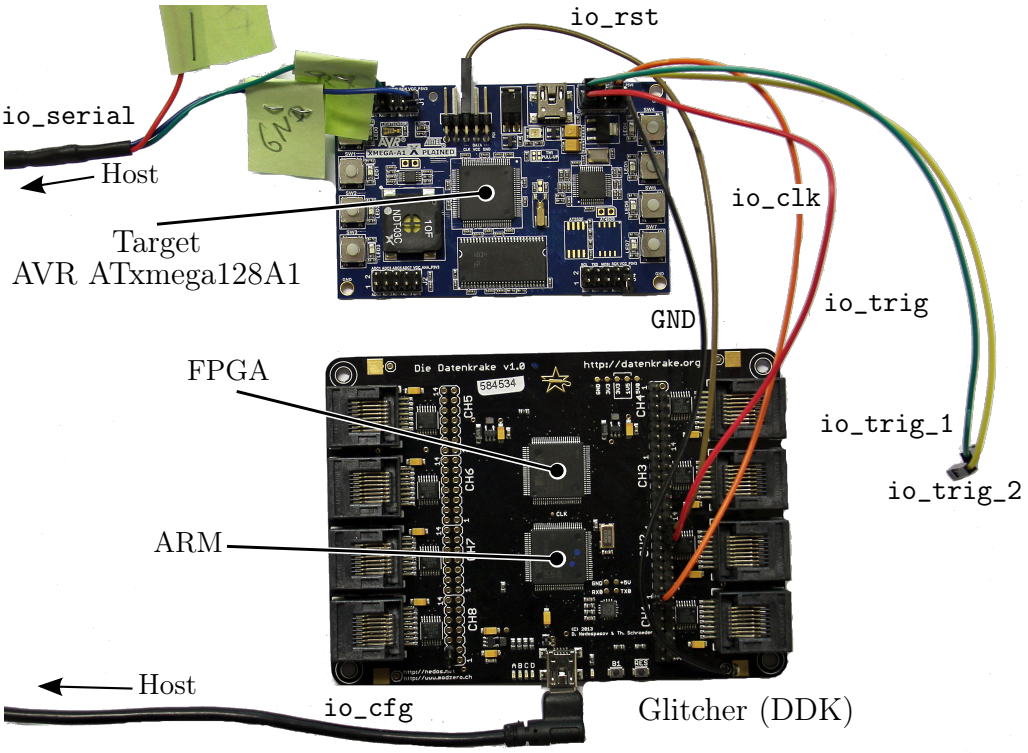

Figure 8.2.: Picture of our practical setup: At the top, we see the target, an AVR ATxmega128A1 as part of an Xplained evaluation board. At the bottom, we see the glitcher, a DDK that consists of an FPGA and an ARM controller. The glitcher provides the CPU clock of the target at `io_clk`. The host that configures the glitcher via `io_cfg` and evaluates the output of the target at `io_serial` is not shown. The unconnected signals `io_trig_1` and `io_trig_2` are only available on profiling devices.

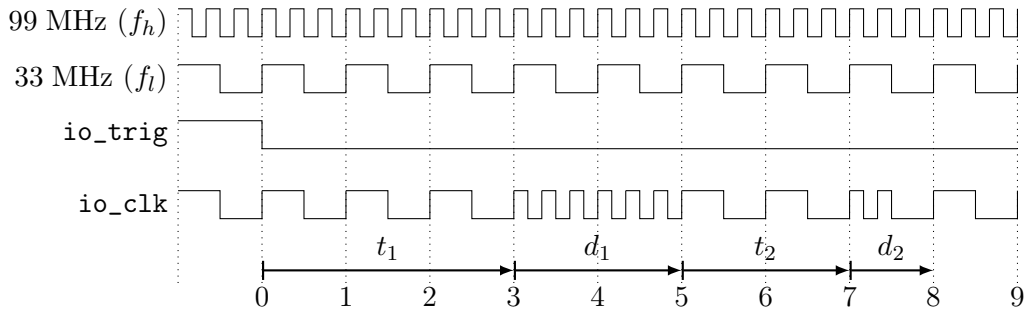

Figure 8.3.: The figure shows the output `io_clk` of the glitcher with two glitches. The first glitch is introduced with a delay of $t_1 = 3$ cycles of the 33 MHz clock, measured relative to the trigger `io_trig`. Its duration is $d_1 = 2$. With $p_1 = 1$, the 99 MHz clock is directly used to generate the glitch pattern. The second glitch is introduced with a delay of $t_2 = 2$ cycles of the 33 MHz clock, measured relative to 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.

The glitcher provides a trigger input `io_trig` for synchronization with the target. Internally, this input is basically used to reset the timer. The main functionality of the glitcher is to generate a clock signal `io_clk` for the target. This output can be switched between f_l and f_h. 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. The glitcher implements 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.

For our second order attacks we have to perform two synchronized glitches. Therefore, we use a queue that is implemented by the glitcher. This queue can be filled with up to 256 triples $(t_1, d_1, p_1), \ldots, (t_{256}, d_{256}, p_{256})$. 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. To fill the queue, the glitcher's internal ARM CPU listens to the serial input at `io_cfg`.

Figure 8.3 shows an example with two glitches. The first glitch is introduced with a delay of $t_1 = 3$ cycles of the 33 MHz clock, measured relative to the trigger `io_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 relative to 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.

8.2.2. Host system

The host is a Linux-based system that configures the glitcher and automates the setup. It provides two serial IO lines. One is used to configure the glitcher while the other is used to receive the output from the target. The host system includes a Python [Pyt14] library to interface with the glitcher. For example, this allows an in-place analysis, e.g., based on a CAS like Sage [Sag14], and logging of the target's output, followed by a direct reconfiguration for the next attacks. Another functionality provided by the host is to periodically execute a self-test routine for testing the functionality of the setup.

8.2.3. Target

For an automated reset of the target the glitcher controls the target's reset pin `io_rst`. Furthermore, the CPU of the target device is clocked with an external clock. We control the CPU clock by connecting the target's clock pin to the glitcher output `io_clk`.

For our concrete attacks, the target generates a trigger on output `io_trig` before the computation of the target program is started. This signal is used to synchronize the target with the glitcher. Generating the trigger on the target is used to simplify the setup. In a real attack, it has to be generated by other means. For example, it could be derived from sniffing the targets IO to locate the command that initiates the attacked computation. Finally, the IO of the target `io_serial` is connected to the host for initiating the attacked computation and for analysis of the computation's results.

For profiling, we use a special target software that provides additional triggers `io_trig_1` and `io_trig_2`. These signals can be connected to an oscilloscope in order to determine the timing of glitches. For more details, we refer to the next section.

8.3. General strategy for second order fault attacks

To configure the glitcher from Section 8.2.1 in a concrete second order attack, the timings t_1, t_2, the duration d_1, d_2, and the patterns p_1 and p_2 of the glitches are required. Here, t_1 and t_2 may depend on the target's secret key. For example, in a pairing computation that uses Algorithm 3.4 to compute $\mathrm{mil}_{r,P}(Q)$ with secret argument Q, all computations of $T_R(Q)$ and $L_{P,R}(Q)$ prior to the first glitch depend on Q. Hence, the timings are a priori unknown to us, which makes it challenging to determine t_1 and t_2. Thus, we split the attack into two steps: profiling phase and target phase. We use the profiling phase to find reasonable configurations (t_1, d_1, p_1) and (t_2, d_2, p_2) for the two glitches. Therefore, we assume that we have access to a profiling device that is similar to the target device. Based on the results of profiling with the profiling device, we attack the target device in the target phase.

We emphasize that once the profiling is completed, we do not need to repeat it when we attack target new samples.

8.3.1. Profiling Phase

The profiling relies on two assumptions:

1. The assembly code of the target implementation is known to us.
2. We are able to execute arbitrary *profiling code* on a *profiling device* similar to the target device.

Based on these assumptions, we first execute a profiling implementation on the profiling device. For the profiling implementation, we modify the target implementation in the following ways:

- We implement triggers `io_trig_1` and `io_trig_2` on two external IO pins (cf. Figure 8.1). Here, `io_trig_1` is raised immediately before the first target instruction and `io_trig_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 a successful skip of the first target instruction.

These modifications allow us to determine t_1 and t_2, the timings of the two target instructions, for every computation of the profiling implementation. Note that t_2 is measured relative to t_1. To measure t_2 we use the emulation mode because we are interested in the delay for the case where the first fault has been successful.

We execute the profiling implementation for different secret keys, chosen uniformly at random from the domain of the secret key, i.e., from $\mathcal{G}_1$ or $\mathcal{G}_2$ for an attack on pairings and from $\mathbb{Z}/r\mathbb{Z}$ for an attack on ECSM. As result, we obtain distributions for t_1 and t_2. Since these distributions are obtained over the random choices of the secret, we will choose t_1 and t_2 in the target phase accordingly.

This step 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 two positive edges at `io_trig` and `io_trig_1`, and between the positive edges at `io_trig_1` and `io_trig_2`. We give a concrete example in Section 8.5.3.

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. Therefore, we perform a large number of experiments where we use the glitcher to introduce glitches shortly after edges at `io_trig_1` and `io_trig_2` that are close to the target instructions. We use the fact that we know the selected secret keys 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.

8.3.2. Target Phase

In the subsequent target phase, the actual target device with the unmodified code and the unknown secret is attacked. 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 precedence based on the results of the profiling phase.

8.4. Target device and implementation

In this section, we give background information on the target hardware, an AVR ATxmega128A1 controller [Atm13], and the target software, the RELIC toolkit [AG], that we used for our practical realizations.

8.4.1. Hardware: AVR ATxmega128A1

We used an AVR ATxmega128A1 controller for several reasons:

- *Toolchain support:* With the avr-gcc compiler and the AVR ATxmega128A1 Xplained[1] evaluation board, free or low-cost products are available.
- *Community support:* The AVR architecture is popular and enjoys large support by the community.
- *Academic significance:* The AVR architecture has been used in academia for many proof of concept realizations of attacks, e.g., [SH08; KQ07; BMH13].
- *Reference setup:* Our clock glitching setup from Section 8.2 has been developed based on the AVR ATxmega128A1 in [Gom14]. Furthermore, the effects of clock glitches on an AVR controller were already analyzed in [BGV11].

The AVR ATxmega128A1 is a controller of the Atmel AVR family with 128 kB of flash memory and 8 kB of main memory. The AVR architecture is a reduced instruction set computer (RISC) architecture with 8 bit and 16 bit instructions. It has a register file with 32 registers `R0...R31`, each of 8 bit width. For even m and $n = m+1$ we write `Rn:Rm` to denote a word register that is composed of the adjacent byte registers `Rn` and `Rm`. The maximum specified CPU frequency is 32 MHz. Note that this is well below 99 MHz, the frequency of the clock glitches that our setup from Section 8.2 generates. In Table 8.1 we list the AVR instructions that are required to follow the details of our attacks.

For more AVR ATxmega128A1 specific details we refer the reader to the device data sheet [Atm13] and the manual [Atm12]. For details on the instruction set, we refer to the AVR instruction set manual [Atm14].

[1] `www.atmel.com/xplain`

Mnemonic	Operand	Description	Operation
`breq`	k	branch if equal	`if z then` $\texttt{PC} \leftarrow \texttt{PC} + k + 1$
`call`	k	call subroutine	$\texttt{PC} \leftarrow k$
`ld`	`Rd`, `Z`	load indirect	$\texttt{Rd} \leftarrow (\texttt{Z})$
`ldd`	`Rd`, `Y`+q	load indirect with displacement	$\texttt{Rd} \leftarrow (\texttt{Y} + q)$
`movw`	`Rd`, `Rr`	copy register pair	$\texttt{Rd+1} : \texttt{Rd} \leftarrow \texttt{Rr+1} : \texttt{Rr}$
`pop`	`Rd`	pop register from stack	$\texttt{Rd} \leftarrow \texttt{STACK}$
`push`	`Rr`	push register on stack	$\texttt{STACK} \leftarrow \texttt{Rr}$
`ret`		subroutine return	$\texttt{PC} \leftarrow \texttt{STACK}$
`rjmp`	k	relative jump	$\texttt{PC} \leftarrow \texttt{PC} + k + 1$
`sbc`	`Rd`,`Rr`	subtract with carry	$\texttt{Rd} \leftarrow \texttt{Rd} - \texttt{Rr} - \texttt{C}$
`sbci`	`Rd`,K	subtract immediate with carry	$\texttt{Rd} \leftarrow \texttt{Rd} - K - \texttt{C}$
`subi`	`Rd`,K	subtract immediate	$\texttt{Rd} \leftarrow \texttt{Rd} - K$

Table 8.1.: Excerpt of the AVR instruction set manual [Atm14]. Here, `PC` denotes the program counter, `STACK` denotes the top of the stack, `z` denotes the zero flag, `C` denotes the carry flag, `Rn` denotes the n-th CPU register, `Y` denotes `R29`:`R28`, `Z` denotes `R31`:`R30`, `(R)` denotes the address stored in the register `R`, k denotes a constant address, and K denotes constant data.

8.4.2. Software: RELIC toolkit and avr-gcc

For the concrete target implementations we used the RELIC toolkit [AG]. It includes `C` implementations of finite field arithmetic, ECC, and PBC for different hardware platforms like Atmel's AVR family. The RELIC toolkit has also been used in TinyPBC for the implementation of PBC in wireless sensor networks [Oli+11]. To the best of our knowledge it is the only freely available implementation of PBC for AVR CPUs. It implements the IBE scheme from Definition 3.14, the BLS signature scheme from Definition 3.15, as well as various pairings. The RELIC default configuration for the AVR ATxmega128A1 is the Eta pairing from [Bar+07].

For our attacks, we use RELIC version 0.3.5 without modifications of the source code. We compiled the library with the avr-gcc-4.x toolchain and optimization level `-O1`. For optimization level `-O2` we did not observe any performance improvements. Nevertheless, we will come back to optimization level `-O2` in Section 8.5.4 because it motivates our extended attack on the final exponentiation from Section 6.2.

To understand the assembly code that we list in the following subsections, it is useful to understand the avr-gcc calling conventions (see, for example, [GCC15]). We do not need to understand all details and restrict to the case where 18 registers suffice to pass all arguments of a function. In avr-gcc, pointers are of width 16 bit. Furthermore, for all cases of our interest, avr-gcc passes arguments of functions via CPU registers. Assume a function `func` with arguments $a_1, \ldots, a_n$. Then the

```
void pb_map_etats(fb4_t r, const eb_t p, const eb_t q) {
        pb_map_etats_imp(r, p, q);
        etat_exp(r, r);
    }
```

Table 8.2.: `pb_map_etats`: RELIC's top level implementation of the Eta pairing on arguments P and Q that are referenced by pointers `p` and `q`, respectively. The results in $\mathbb{F}_{q^4}$ is referenced by the pointer `r`. Note that the exponentiation `etat_exp` uses the same memory for its argument as well as for the result. The memory is referenced by the pointer `r`.

arguments are individually padded to a multiple of 16 bit and passed in registers `R25`, `R24`, ..., `R8`, starting with the MSBs of a_1 at `R25`. The result is returned in `R25`, `R24`, ..., `R18`. For example, for the function with interface

```
int func(int *a, int *b, int *c);
```

the pointer `a` will be passed in `R25:R24`, the pointer `b` will be passed in `R23:R22`, and the pointer `c` will be passed in `R21:R20`. The result will be returned in `R25:R24`.

8.5. Second-order faults on the pairing computation

This section describes our concrete second order fault attack from Section 6.2 on the protocol Pair-Argument (cf. Definition 3.17). In Section 8.5.1 we give details on the concrete implementation of pairings on our target. In Section 8.5.2 we explain the concept of or basic attack from [Blö+14] and we identify the target instructions of the attack. In Section 8.5.3 we give details on the concrete execution of the attack, including profiling phase and target phase. In Section 8.5.4 we explain how the extended attack that we analyze in Section 6.2 can be realized on our target implementation.

8.5.1. Implementation of pairings on our target

The RELIC default configuration for AVR devices defines the Eta pairing $\eta : E(\mathbb{F}_q) \times E(\mathbb{F}_q) \to \mathbb{F}^*_{q^k}$ from Section 6.2.1 as the standard pairing. It is implemented by the function `pb_map_etats` that is outlined in Table 8.2. The pairing on inputs P and Q is computed in two steps: Miller algorithm and final exponentiation. The Miller algorithm is implemented by the function `pb_map_etats_imp` similar to Algorithm 6.1. The main loop of the Miller algorithm is implemented by a `for` loop that increments a loop counter in each iteration. The final exponentiation is implemented by the function `etat_exp` similar to Algorithm 6.2.

In our experiments both arguments P and Q are loaded from the internal memory. Then $\eta(P, Q)$ is computed on the target device and the output is returned on the serial IO `io_serial`. We remark that loading the public argument from memory

and not via the serial line `io_serial` helps to simplify the setup, but is not essential for the attack.

8.5.2. Concept of the attack

We start with a high level outline of the attack before we go into the details of the assembly code.

Outline

We attack an instantiation of the protocol Pair-Argument-1 from Definition 3.17. Here, we are allowed to choose the first argument P of the pairing and the second argument Q of the pairing is secret. According to the definition of Pair-Argument, we attack an isolated pairing computation, including Miller algorithm and final exponentiation. We use two faults to attack the pairing. We use the first fault to terminate the loop of Algorithm 6.1 after the first iteration. With the second fault, we completely skip the final exponentiation.

First, we choose an arbitrary point $P \in E(\mathbb{F}_q)$ with $P \neq \mathcal{O}$ and compute $\eta(P, Q)$ with secret Q on the target device. During the computation of $\text{mil}_{n,P}(Q)$ we introduce a fault in Line 12 of Algorithm 6.1. This fault skips the instruction that jumps to the beginning of the loop. Hence, the loop is terminated after one iteration. Then we introduce a second fault to skip the complete final exponentiation of Algorithm 6.2. If both faults are successful, the target device returns the state of the variable b in Algorithm 6.1 at the end of the first iteration. With b, we recover Q based on the analysis of Section 6.2.

Target instructions

To understand how we attack the `for` loop, we refer to Table 8.3. It shows how the compiler generates the end of the `for` loop. In Line 5 and Line 6 the loop counter is decremented by 1. If the counter reaches 0, the `zero` flag of the CPU will be set. In Line 7 the CPU branches over the `rjmp` instruction if the `zero` flag is set. If the `zero` flag is not set, the CPU jumps to the beginning of the loop in Line 8. An instruction skip fault that removes the `rjmp` instruction in Line 8 causes the loop to terminate immediately and proceed with the code following the loop in Line 10.

Now we explain how we skip the final exponentiation. Therefore, we refer to RELIC's implementation of the Eta pairing shown in Table 8.2. The variable `r` is a pointer to the memory that will store the argument and the result of the exponentiation. Hence, `etat_exp` will overwrite the memory at `r`. If the call to the function `etat_exp` is skipped, the memory will not be overwritten and it will contain the erroneous result of the Miller algorithm from `pb_map_etats_imp`.

Table 8.4 lists the corresponding assembly of the function `pb_map_etats` that implements the Eta pairing. In Line 6 and Line 7 the registers `R23`:`R22` and `R25`:`R24` are initialized with the pointer `r`. Then, in Line 8 the computation of the

```
2  ...
3   call fb4_mul_dxs        ; call sparse multiplication
4  .LVL43:
5   subi r16,1              ; decr. LSB of loop counter
6   sbc r17,__zero_reg__    ; update MSB with carry
7   breq .+2                ; exit loop if zero
8   rjmp .L2                ; jump to beginning of loop
9  .LBE2:
10  subi r28,36             ; clean stack
11  sbci r29,-2             ; -"-
12  out __SP_L__,r28        ; -"-
13  out __SP_H__,r29        ; -"-
14  pop r29                 ; -"-
15 ...
```

Table 8.3.: Assembly of Miller algorithm with conditional branch at the end of the `for` loop. The assembly corresponds to Line 11–Line 12 of Algorithm 6.1 and was generated with avr-gcc. In our attack, we skip the `rjmp` instruction in Line 8.

```
1  pb_map_etats:
2   push   r28             ; save context
3   push   r29             ; -"-
4   movw   r28, r24         ; init pointer to result r
5   call   pb_map_etats_imp; call Miller algorithm
6   movw   r22, r28         ; init pointer to base
7   movw   r24, r28         ; init pointer to result
8   call   etat_exp         ; call exponentiation
9   pop    r29              ; restore context
10  pop    r28              ; -"-
11  ret                     ; return
```

Table 8.4.: Assembly of the Eta pairing (`pb_map_etats`) with function call to Miller algorithm (`pb_map_etats_imp`) and function call to final exponentiation (`etat_exp`). The code was generated by avr-gcc. In our attack, we skip the `call` in Line 8.

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 8.5.: Distribution of the execution time t_1 of the `rjmp` instruction in Table 8.3, depending on the input Q of Algorithm 6.1.

final exponentiation is called. By skipping this call, we completely skip the final exponentiation.

Remark 8.1. Note that we can also virtually skip the final exponentiation by redirecting the pointer `r` (cf. Section 8.1.3). If we introduce an instruction skip at Line 7, we skip the initialization of the pointer `r`. Then, the register `R25:R24` will not contain the address of the variable b from Algorithm 6.1. The subsequent execution of `etat_exp` will therefore update memory at a different location. This has the effect of virtually skipping the final exponentiation. This approach might circumvent simple countermeasures that try to ensure the execution of `etat_exp`.

8.5.3. Execution of the attack

According to Section 8.3 we split the attack into two phases, profiling phase and target phase that we explain in the following.

Profiling Phase

As explained in Section 8.3.1 we start with a profiling to estimate the timings t_1 and t_2 of the target instructions. In a first step, we estimated t_1, the clock cycle of the `rjmp` instruction in Line 8 of Table 8.3 relative to the trigger at `io_trig` that is set at the beginning of the pairing computation. Therefore, we executed approximately 35 000 experiments with random choices of Q and measured t_1 for each experiment. The distribution of t_1 is given in Table 8.5.

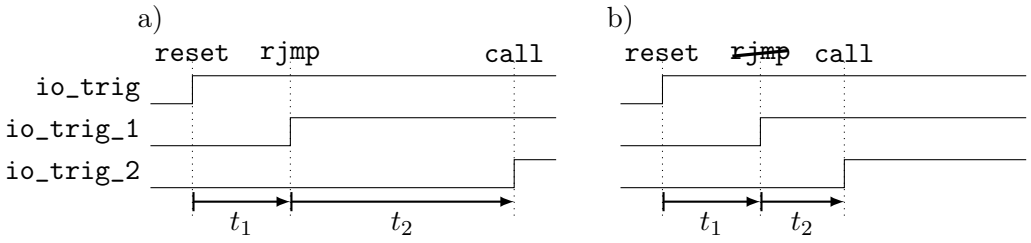

Figure 8.4.: Schematic illustration of the timings of target instructions in profiling mode. In a), we see an execution of the pairing without faults. We use `io_trig` and `io_trig_1` to estimate the timing t_1 of the `rjmp` instruction. In b), we see the execution of the pairing where we skip over the `rjmp` instruction in profiling mode. The timing t_2 from trigger `io_trig_1` to `io_trig_2` decreases because the `for` loop is only executed for one iteration.

Then we determined t_2, the number of clock cycles that pass between the targeted `rjmp` and `call` instructions. To estimate this value for the situation where the first fault has been successful we use our emulation mode of the profiling code. It emulates the first fault by skipping over the `rjmp` instruction at t_1. We obtained a constant value of $t_2 = 28$. Here, t_2 is constant because if the first glitch was successful in leaving the `for` loop, we see from Algorithm 6.1 and Table 8.2 that the code executed between the `rjmp` at t_1 and the `call` at t_2 is independent of the secret.

In Figure 8.4, we illustrated two measurements in profiling mode. In Figure 8.4 a), the target generates triggers on `io_trig_1` and `io_trig_2` immediately before executing the target instructions. This allows us to measure t_1 relative to the start of computation at the edge of `io_trig`. In Figure 8.4 b), we use the emulation mode to jump over the `rjmp` instruction for simulating an instruction skip. This allows us to measure t_2 for the case where the loop is terminated successfully.

To select combinations of d_1, d_2, p_1 and p_2 of glitch width and glitch pattern we injected approximately 40 000 faults. This took us less than 72 hours. Since we know Q during profiling, and hence also the values of t_1 and t_2, we are always able to introduce the faults at the correct instructions. Regarding the two patterns p_1 and p_2 depicted in Figure 8.3, both showed a high success rate. For the duration of the glitches, we found that $d_1 = 3$ or $d_1 = 5$ and $d_2 \leq 5$ gave the best results.

Target Phase

In the attack on the target device with unknown Q, we scheduled the timing t_1 of the first glitch according to the frequency of occurrence shown in Table 8.5. We started our experiments with $t_1 = 582\,200$ clock cycles and worked towards delays that occurred less often during profiling.

We combined each value of t_1 with each combination of $d_1 \in \{3, 5\}$, $d_2 \in [1, 5]$,

and $p_1, p_2 \in \{1, 2\}$ that we determined in the profiling phase. For t_2 we added a small safety margin to the value $t_2 = 28$ from profiling such that we used $t_2 \in [26, 30]$ in the attack. Furthermore, we repeated each combination for 10 times because even with the correct parameters, glitching is not always successful. Hence, for each value of t_1 we scheduled $2 \cdot 5 \cdot 2 \cdot 2 \cdot 5 \cdot 10 = 2\,000$ experiments. For our setup, one test requires 7.5 seconds on average. This includes configuration of the glitcher, communication from target to host, pairing computation, and self-tests. Hence, we are able to perform more than 10 000 experiments per day.

As we show in Section 6.2 we are able to compute the secret Q from the target's output if both instruction skips were successful. Furthermore, we can verify candidate secrets based on the result of a correct pairing computation. This allows us to recognize the original secret and hence the first successful attack.

We repeated the attack for five different secrets, drawn uniformly at random from $E(\mathbb{F}_q)$. We were always successful in skipping both instructions with at least one of the selected combinations. The analysis of the experiments showed that for all secrets it occurred that t_1 was either 582 200 or 582 101. This is in line with the distribution in Table 8.5. Hence, for each attack we required at most $2 \cdot 2\,000$ experiments. Due to the concentration of the distribution of t_1 at 582 200, typically much fewer experiments were required and it took us only minutes to be successful.

8.5.4. Extended attack on the final exponentiation

Now we present an extension of our attack for the case where the RELIC library is compiled with optimization level `-O2`. This is an example where it is not possible to completely skip the final exponentiation. Here, the reason is that the compiler replaces the `call` in Line 8 of Table 8.4 by completely inlining the function `etat_exp`.

It is still possible to attack the pairing at the cost of a slightly more complex analysis. As explained in Section 6.2, the final exponent is given as $e = \left(q^2 - 1\right)\left(q - \sqrt{2q} + 1\right)$ for the attacked Eta pairing. With regard to our analysis, the RELIC implementation of the finial exponentiation is similar to Algorithm 6.2. Now, in our extended attack, we do not use the second fault to skip the complete final exponentiation. Instead, we use the second fault to skip the multiplication $v_2 \leftarrow v_1 \cdot v_2$ of $b^{(q^2-1)}$ with $b^{(q^2-1)q}$ in Line 6 of Algorithm 6.2. This eliminates the term $(q^2 - 1)q$ from the exponent and finally results in a modified exponent $\tilde{e} = \left(q^2 - 1\right)\left(-\sqrt{2q} + 1\right)$.

To see that it is actually possible to skip this particular multiplication for the RELIC implementation, we listed parts of the assembly code that correspond to Algorithm 6.2 in Table 8.6. It corresponds to Line 5 and Line 6 of Algorithm 6.2. In Line 6 of Table 8.6, the Frobenius automorphism of b^{q^2-1} is computed that implements exponentiation with q. Then, in Line 7 up to Line 9, the register `R21:R20` is initialized with a pointer to the multiplicand $v_2 = b^{q^2-1}$. In Line 10, the register `R23:R22` is initialized with a pointer to the multiplier $v_1 = b^{(q^2-1)q}$. In Line 11, the pointer to v_2 is copied into `R25:R24`. This initializes the pointer that

```
1   ...
2   movw    r22, r28    ; init source pointer v_2
3   subi    r22, 0x77   ; -"-
4   sbci    r23, 0xFF   ; -"-
5   movw    r24, r8     ; init destination pointer v_1
6   call    fb4_frb     ; call frobenius map v_1<-v_2^q
7   movw    r20, r28    ; init pointer to multiplicand v_2
8   subi    r20, 0x77   ; -"-
9   sbci    r21, 0xFF   ; -"-
10  movw    r22, r8     ; init pointer to multiplier v_1
11  movw    r24, r20    ; init pointer to product
12  call    fb4_mul     ; call multiplication of v_1*v_2
13  ...
```

Table 8.6.: Assembly snippet of the final exponentiation (`etat_exp`), generated with avr-gcc and optimization level `-O2`. This part of the assembly corresponds to Line 5 and Line 6 of Algorithm 6.2. In our attack, we skip the `call` in Line 12.

stores the product. Hence, the multiplication function `fb4_mult` overwrites the multiplicand v_2. In Line 12, the multiplication $v_2 \leftarrow v_1 \cdot v_2$ is executed. Skipping the `call` to this multiplication prevents the update of v_2 that stores the value b^{q^2-1}. This has the effect of removing the term $(q^2-1)q$ from the exponent to obtain the exponent $\tilde{e}$.

8.6. Singular curve attack on BLS signatures

In this section, we give background information on the practical realization of our fault attack from Section 7.2 on BLS short signatures (cf. Definition 3.15). As explained in Section 3.3.3, BLS signatures are an example for the scheme Hash-And-Multiply from Definition 3.18. We perform a second order attack. We use the first fault to obtain a point on a singular curve. With the second fault, we circumvent the point validation countermeasure (see Section 7.5) of our implementation.

In Section 8.6.1 we give details on the concrete implementation of BLS signatures on our target. In Section 8.6.2 we explain the concept of the attack and we identify the target instructions of the attack. In Section 8.6.3 we give details on the concrete execution of the attack, including profiling phase and target phase.

8.6.1. Implementation of BLS signatures on our target

The RELIC library already provides an implementation of BLS signatures. According to Definition 3.15, to sign a message M, the target device with secret key s computes $P = \text{HashToCurve}(M)$, $Q = sP$, and returns $\sigma = Q$ as a signature under message M. In RELIC, HashToCurve is implemented similarly to Algorithm 3.3 by

the function `ep_map`. This function uses the function `ep_rhs` as a sub-program to compute the right hand side $x^3 + a_4x + a_6$ of (2.3) in a way similar to Line 2–Line 5 of Algorithm 3.1. The function `ep_rhs` itself uses the function `fp_add_dig` to implement the addition of $x^3 + a_4x$ and a_6 in Line 5 of Algorithm 3.1. Furthermore, the implementation of ECSM does not use the parameter a_6 and hence, the implementation is IPV (cf. Definition 7.1).

As a basic protection against fault attacks, we extended the implementation with the point validation countermeasure that was discussed in Section 7.5.1: Before the signature σ is released, the implementation calls the function `check_point_valid` to check that Q is a valid point in the sense of Definition 3.3. If it is valid, the device outputs $\sigma = Q$. If not, the device responds with an error message. Hence, following Definition 7.1, our implementation is IPV2. The check is implemented with an `if` statement and with no special protection against attacks.

To instantiate the corresponding groups $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_T$ of the scheme we use BN curves from Theorem 3.6. For the concrete parametrization, we refer to Appendix A.1. Note that BN curves have j-invariant 0 and hence, we can apply the attack from Section 7.2 on Hash-And-Multiply. BN curves are natively supported by RELIC.

In our experiments the input message M is loaded from the internal memory of the target. Loading the message from memory and not via the serial line helps to simplify the setup, but is not essential for the attack. After the signature under M has been computed on the target device, it is returned on the serial IO `io_serial`.

8.6.2. Concept of the attack

First, we give an outline of how we realize the attack. Then we show the concrete instructions that we attack.

Outline

Recall that we use the first fault to skip the addition with a_6 in Line 5 of Algorithm 3.1. With the second fault, we eliminate the point validity check.

First, we choose a message M and start the signature computation for M. For a proper selection of M we refer to Section 7.2 and more specifically to the example in Appendix A.1. While the target computes $P = \text{HashToCurve}(M)$ with Algorithm 3.3 as part of the signature generation, we introduce the first instruction skip fault. More specifically, we place the fault within `ep_rhs` that corresponds to the addition with a_6 in Line 5 of of Algorithm 3.1. This results in $\tilde{P}$ on the singular curve $\tilde{E}$. Because BN curves have prime order r the cofactor is $c = 1$ and Algorithm 3.3 outputs $\tilde{P}$. Then, the target computes a faulty signature $\tilde{Q} = s\tilde{P} \in \tilde{E}$.

Finally, the target checks the validity of $\tilde{Q}$. With the second instruction skip fault, we remove a branch instruction of the check. The effect is that both, the `if` and the `else` branch are executed such that the device outputs the invalid point $\tilde{Q}$.

```
1  .ep_rhs:
2   ...
3   movw    r30, r24        ; init pointer to a_6
4   ld      r20, Z          ; load a_6 into R20
5   movw    r22, r28        ; init pointer to v=x^3+a_4x
6   subi    r22, 0xEB        ; -"-
7   sbci    r23, 0xFF       ; -"-
8   movw    r24, r22        ; copy pointer for result
9   call    fp_add_dig      ; call addition of v and a_6
10  rjmp    .+18
11  ...
```

Table 8.7.: Assembly snippet of point decompression, generated with avr-gcc. This part of the assembly corresponds to Line 5 of Algorithm 3.1, i.e., to the addition of $x_P^3 + a_4 x_P$ and a_6. In our attack, we skip the `call` in Line 9.

Hence, a successful attack will provide us with the DLOG instance $\tilde{Q}, \tilde{P} \in \tilde{E}$ that we can analyze based on Section 7.2.

Target instructions

To understand how we attack the addition with a_6, we refer to Table 8.7. It shows the avr-gcc generated assembly code of our RELIC based implementation. Concretely, this part corresponds to Line 5 of Algorithm 3.1 on input $(x_U, b) = \mathrm{H}\left(M. \lceil 0 \rceil_2^t\right)$. Hence, with $a_4 = 0$ it will add $v = x_U^3$ and a_6. In Line 3 of Table 8.7, a pointer to a_6 is loaded into the pointer register `Z` (`R31:R30`). Then, the value of a_6 is loaded into the register `R20`. In Line 5–Line 7 the register `R23:R23` is initialized with a pointer to $v = x_U^3$. In Line 8 the pointer is copied to register `R25:R24`. Then the function `fp_add_dig` is called that adds the value of a_6 in `R20` to the variable that is referenced by the pointer in `R23:R22`. The result is written to the address in `R25:R24`. Because both, `R23:R22` and `R25:R24` point to the same variable v, the function `fp_add_dig` overwrites v with $v + a_6 = x_U^3 + a_6$.

Now, an instruction skip fault that removes the `call` instruction in Line 9 prevents an update of v that is referenced by `R25:R24`. Hence, it virtually skips the addition $x_U^3 + a_6$ in Line 5 of Algorithm 3.1. If the message M was chosen properly, x_U is a square, and hence `ep_map` that corresponds to HashToCurve will output $\tilde{P} = (x_U, x_U\sqrt{x_U}) \in \tilde{E}$.

Remark 8.2. Note that there are various other instructions that we can skip to remove the addition with a_6. For example, we can redirect the target pointer in `R25:R24` with the effect that a_6 is added to data at another address (cf. Section 8.1.3) that does not effect the subsequent computation. A further possibility in our case, where $a_6 = 17$ (see Appendix A.1) fits into one word, is to skip the actual arithmetic `add` instruction within `fp_add_dig`. In fact, we found many different instruction

```
 1  ldd     r24,Y+5            ; init pointer to signature
 2  ldd     r25,Y+6            ; -"-
 3  call    check_point_valid  ; check if point is valid
 4  std     Y+3,r24            ; store outcome of check
 5  std     Y+4,r25            ; -"-
 6  ldd     r24,Y+3            ; load outcome of check
 7  ldd     r25,Y+4            ; -"-
 8  sbiw    r24,0              ; subtract/compare with 0
 9  brne    .L40               ; branch if valid
10  ...
11  call    my_printf          ; return error message
12  pop     __tmp_reg__
13  pop     __tmp_reg__
14  rjmp    .L41               ; jump over signature out
15 .L40:
16  ...
17  ldd     r24,Y+5            ; init pointer to signature
18  ldd     r25,Y+6            ; -"-
19  call    print_ep_element   ; return signature
20  ...
```

Table 8.8.: Assembly of point validity check, generated with avr-gcc. If the result of ECSM referenced by `R25:R24` is a valid point, it is returned as the signature in Line 19. Otherwise, an error message is returned in Line 11. In our attack, we target the `rjmp` instruction in Line 14.

cycles for the targeted implementation where a fault had the effect of skipping the addition with a_6.

With the second fault, we circumvent the point validity check. The listing in Table 8.8 shows the assembly that calls the check and evaluates its outcome. In Line 1 and Line 2 the register `R25:R24` is initialized with a pointer to the result $Q = sP$ of the ECSM. Then, in Line 3 the function `check_point_valid` that checks the validity of Q is called. This function returns 0 for invalid points and 1 for valid points. The result is returned in `R25:R24`. Then, the subtraction in Line 8 realizes a comparison with 0. If and only if the value of `R24` is 0, the `sbiw` instruction sets the `zero` flag. Hence, in the case of a valid point, where `R24` is not equal to 0, the conditional branch in Line 9 will branch over the error handling. In the case of an invalid point, where `R24` is equal to 0, no branch occurs and the code following line Line 10 is executed, including the output of an error message in Line 11. In Line 14, the code jumps over the branch that handles valid points. Hence, in case of a valid point, in Line 17 and Line 18, `R25:R24` is initialized with a

pointer to the signature $\sigma = Q$. Finally, in Line 19 the signature is returned.

With our instruction skip fault, we target the `rjmp` instruction in Line 14 for an invalid point. If we are successful both branches are executed and the signature is also returned for invalid points.

Remark 8.3. There are various options to circumvent the validity check. Instead of skipping the `rjmp` instruction in Line 14, it is also possible to skip the `sbiw` instruction in Line 8. The reason is that in our case, the `zero` flag is not set at Line 6. If we skip the `sbiw` instruction, even for invalid points the `zero` flag will not be set. Hence, in Line 9 we branch to the output of the signature.

We conclude that it is difficult to identify and protect vulnerable instructions and that a validity check has to be implemented carefully to be effective. This is especially true because for both faults, we had several options for choosing the target instructions of our instruction skip faults.

8.6.3. Execution of the attack

As described in Section 8.3 we split the attack into profiling phase and target phase.

Profiling phase

To estimate the timings t_1 and t_2 of the target instructions, we perform a profiling (cf. Section 8.3.1). During the profiling, we compute a signature on the profiling device under the same message M that we later use in the attack on the target device.

To learn t_1, the timing of the `call` instruction in Line 9 of Table 8.7, we use the profiling code described in Section 8.3.1. The profiling code sets a trigger `io_trig_1` before the target `call` instruction. This allows us to measure the number of clock cycles between the triggers `io_trig` to `io_trig_1` on the profiling device. Later, we can use the estimate for t_1 from the profiling also for the target device. The reason is that t_1 is independent of the secret s because s is only used in the ECSM after the target `call` instruction. Therefore, we do not need to measure the distribution of t_1 over the random choices of the secret as we do it in Section 8.5.3 for an attack on the pairing computation.

To estimate the timing t_2 of the second glitch we directly use the target device with the target key. We compute a signature for M on the target device. During the computation, we introduce a fault at our estimate for t_1 that we obtained before. From the target's response, we measure the time that it takes to compute a signature and to perform the validity check. If the target responds with an error message, we assume that the first fault was successful and that we obtained an invalid point on the singular curve. We see from Table 8.8 that the error message in Line 11 is returned close before the target `rjmp` instruction in Line 14. Hence, the delay between the first glitch and the response of the device gives a good estimate for t_2.

Key No.	t_1	t_2	Experiments
1.	181 270	208 286 993	9
2.	181 270	208 421 212	65
3.	181 270	207 049 930	123
4.	181 270	208 414 882	123
5.	181 270	207 115 540	8
6.	181 270	203 383 894	183
7.	181 270	208 503 182	82
8.	181 270	208 317 830	124
9.	181 270	207 055 821	24
10.	181 270	205 729 894	70

Table 8.9.: Timing results of our attacks on 10 different secret keys. The table lists the timings t_1 and t_2 of the two glitches in instruction cycles. Only the timings of the first successful experiment for the corresponding key are given. The last column shows how many experiments were required to extract this key.

To select combinations of d_1, d_2, p_1 and p_2 of glitch width and glitch pattern (cf. Section 8.2.1), we re-use our findings from Section 8.5.3.

Target phase

In the attack, we tried different values for t_1 and t_2 centered around our estimations from the profiling. If both faults are successful, we obtain a point on the singular curve and we compute the DLOG from it as described in Section 7.2. The DLOG computation on the singular curve is faster than the signature computation on the target and hence, it is not a bottleneck for the attack. This confirms the efficiency of our attack as discussed in Section 7.1.3.

We repeated the attack for 10 different secret keys and were always successful in extracting the key. Table 8.9 shows the timings t_1 and t_2 of the first experiment that allowed us to extract the corresponding secret. We see that the delay of the first glitch is fixed at $t_1 = 181\,270$ instruction cycles. This confirms our observation that the first target instruction does not depend on the secret key s. We also see that t_2 significantly depends on the secret key. Hence, we conclude that the timing of RELIC's ECSM implementation is not independent of the secret scalar s. Note that based on our estimations from profiling, at most 183 experiments were required to extract the key. In summary, the experiments confirm that our attack is a serious threat on unprotected devices and that it has to be considered for fault resistant implementations.

Chapter 9.

Conclusion and future work

In this chapter, we draw a short conclusion based on our results and we present two examples for future work.

9.1. Conclusion

In this thesis, we analyzed the vulnerability of PBC to physical attacks. In particular, we answered some specific questions of Section 1.5 from the beginning of this thesis:

1. Whelan and Scott [WS06] asked if using the secret as the first argument of the pairing offers protection against SCAs. In Chapter 4 we presented new SCAs on the computation of the Tate and the Ate pairing to show that this is not the case.

   Our attacks are based on standard optimizations for the pairing computation. Our attack on the computation of the Tate pairing in affine coordinates exploits that the secret is defined over the base field $\mathbb{F}_q$ while the public argument is defined over the extension field $\mathbb{F}_{q^k}$. Our attack on the Ate pairing in affine coordinates exploits the efficient representation of the secret based on twists. Finally, our attack on the computation in projective coordinates exploits the representation of the secret in mixed projective coordinates.

2. With respect to FAs, the results of ,e.g., [WS07; Las+14] motivate the question how much protection is offered by the two-step computation of the pairing composed of Miller algorithm and final exponentiation. Towards answering this question, we presented a framework to analyze FAs on the complete pairing computation in Chapter 5. Furthermore, in Chapter 6, we presented concrete second order attacks based on instruction skips on the complete pairing computation.

   Our active attacks are based on the same efficient representations of the secret as our passive attacks on the Tate and the Ate pairing in affine coordinates. Furthermore, to attack the final exponentiation we exploit that one factor of the final exponent has a small q-ary representation that also improves the efficiency of implementations.

3. So far, it was not shown that building blocks other than the pairing computation itself offer new vulnerabilities of PBC to physical attacks. To show that this is the case, we presented an attack based on singular curves in Chapter 7 that targets point decompression. Point decompression is a building block of other algorithms, e.g., for hashing strings to points on an elliptic curve in pairing-based signature schemes. Hence, we were able to turn our attack on point decompression into attacks on pairing-based signature schemes.

   Our most powerful attack in the context of point decompression is based on curves with j-invariant 0 that provide especially efficient implementations of pairing-based primitives.

4. Prior to our results, it was not clear if FAs on PBC are really practical. In Chapter 8 we presented the practical realization of our attacks on the pairing computation from Section 6.2 and on point decompression from Section 7.2. Our attacks, with applications in pairing-based encryption schemes and pairing-based signature schemes, demonstrate that physical attacks are a serious threat in the context of PBC.

From this outline, we see that optimized implementations that are based on additional structure of the corresponding groups are often accompanied by additional vulnerabilities. Nevertheless, those efficient implementations are required, especially on resource constrained devices such as smart cards where physical attacks are relevant. Therefore, avoiding efficient representations is not the right conclusion. Especially because we saw that implicit countermeasures, such as placing the secret as the first argument of the pairing or the two-step pairing computation, do not provide a strong protection against attacks.

Hence, we require countermeasures against physical attacks on PBC. Such countermeasures were already proposed, for example, in [PV04; Kim+06; STO08]. But the history of ECC and RSA shows that often a countermeasure against one kind of attacks opens the way for another kind of attacks. We should take the results of this thesis and of related previous work into account before rolling out PBC in scenarios where physical attacks are relevant. We also conclude that in the complex setting of PBC more work has to be done to reach the maturity of standard ECC with respect to resistance against physical attacks. We further recommend to pursue standardization of parameters, algorithms, and schemes to reduce the complexity of PBC and to narrow the playground for physical attacks.

9.2. Future work

Now we give two examples for future work. The first example is related to countermeasures against SCAs on the pairing computation and to SCAs on pairing-based signature schemes. The second example is related to FAs on the pairing computation.

9.2.1. Passive attacks on point randomization and pairing-based signature schemes

We saw in Chapter 4 that the pairing computation is vulnerable to SCAs that use a DPA of finite field operations like multiplication or addition. In Chapter 7 we presented FAs against pairing-based signature schemes. A further question is how we can combine both settings into an SCA on pairing-based signature schemes.

For the type of signature schemes where the secret is a scalar as input to ECSM like in [BLS04b], the situation is similar to standard ECC. Hence, attacks on ECSM like [Cor99] and the corresponding countermeasure also apply here. But we also investigated signature schemes where the secret is a point on the curve [CC03]. Here, the results from standard ECC with respect to ECSM do not directly apply. But this does not mean that schemes like [CC03] are not vulnerable to SCAs. As one indication we consider the attack of [Fou+08b] on the randomization of secret exponents. In [Fou+08b] it is shown that the addition of a fixed multi-precision integer s with a random mask r may leak information about s in an SCA. The idea is to measure the probability of the carry bits that occur in the addition over the random choices of r. In [Fou+08b] this technique is used to attack countermeasures against SCAs that are based on the randomization of the secret scalar s.

As a first step, it would be interesting to analyze the practicability of this attack on the point blinding countermeasure of [PV04]. Here, the bilinearity of the pairing is used to mask the public argument Q before computing the pairing on input P and Q:

$$\mathrm{e}(P, Q) = \mathrm{e}(P, Q + R) / \mathrm{e}(P, R).$$

From the Miller algorithm (cf. Algorithm 3.4 and Definition 3.8) we see that the x-coordinates of P and R are subtracted during the computation of $\mathrm{e}(P, R)$. With standard representations of finite field elements, we can consider the coordinates also as multi-precision integers. Hence, in an SCA we can query the pairing computation multiple times to apply the attack of [Fou+08b] on the coordinate additions to recover the x-coordinate of P, and hence also P. Because none of the arguments is under control of the attacker, the attack certainly makes stronger assumptions about the SCA than the DPA on finite field multiplications that we used in Chapter 4. Hence, the question is if this attack is a real thread.

In a second step, the technique could be transferred to ECSM of a secret base point P with a random scalar r as it occurs, for example, in the signature scheme of [CC03]. Consider the last iteration of a double-and-add routine for computing rP and furthermore assume that r is odd. In this case, $(r-1)P + P$ is computed. With $R = (r-1)P$ we obtain the addition $R + P$ of a random point R with the secret point P. From the addition formula on elliptic curves (cf. Definition 2.9) we see that the x-coordinates of R and P are subtracted as part of this computation. Hence, we could try to apply the SCA from [Fou+08b] as before to recover P.

9.2.2. Invalid point attacks on the pairing computation

In Chapter 6 we presented FAs on the pairing computation when one argument of the pairing is public and the other argument is secret. In Chapter 7 we presented invalid point attacks where the public input of ECSM is modified into an invalid point to recover the secret scalar. This suggests an attack that combines both approaches into invalid point attacks on the pairing computation where the public argument of the pairing is modified into an invalid point. The question is how we can exploit an invalid public argument, for example, on an elliptic curve with smooth order, on a singular curve, or in a small subgroup of the original curve. The former two cases seem difficult to analyze because the invalid public argument is on a different curve than the secret argument. So we consider the latter case where the public argument is in a small subgroup on the same curve as the secret argument. Let P be the public argument of the pairing in a small subgroup of order c. Furthermore, let Q be the secret argument of the pairing of large order r. Assume a pairing is computed based on the Miller function $\mathrm{mil}_{n,P}$ from Definition 2.31 with divisor $n\,[P] - [nP] - (n-1)\,[\mathcal{O}]$. Let P be of order c with $r = ic + j$ and $j < c$. By comparing divisors we obtain:

$$\mathrm{mil}_{r,P} = \mathrm{mil}_{c,P}^{i}\,\mathrm{mil}_{j,P}\,. \tag{9.1}$$

We know from Chapter 5 that the degree of the involved functions plays a crucial role in the analysis of FAs on pairings. If c is small, then the degrees of $\mathrm{mil}_{c,P}$ and $\mathrm{mil}_{j,P}$ are also small. Hence, the large degree of $\mathrm{mil}_{r,P}$ is absorbed into the exponent i. Therefore, a possible first step to approach invalid point attacks on the pairings would be to answer the question if we can compute Q from $\mathrm{mil}_{r,P}(Q)$ by exploiting the additional structure of $\mathrm{mil}_{r,P}$ from (9.1).

Appendix A.

Numerical examples

A.1. Singular curve attack for j-invariant 0

In this section, we give a numerical example for our attack from Section 7.2 for the parameters that we used in the practical realization of the attack in Section 8.6. We attack a BN curve as defined in Theorem 3.6. Our concrete instantiation of domain parameters is the curve $y^2 = x^3 + 17$ defined over a prime field with characteristic

$$p = 205523667896953300194896352429254920972540065223.$$

The curve has prime order

$$\#E(\mathbb{F}_p) = r = 205523667896953300194895899082072403858390252929$$

and hence cofactor $c = 1$.

For the signed message, we chose the ASCII representation of $m =$ "Hello world!". With the RELIC implementation of the hash function H that is used in HashToCurve (cf. Section 8.6.1), this message results in $\mathrm{H}\left(m. \lceil 0 \rceil_2^t\right) = (x_U, b)$ with $\sqrt{x_U} \in \mathbb{F}_p$. Then with $c = 1$ and $b = 0$, the point $\tilde{P} = (x_P, y_P)$ with

$$\begin{aligned} x_P = x_U &= 178593680027287028005098471379742442193364077343 \\ y_P = x_U\sqrt{x_U} &= 946605451840607112720365459179817908653163345 18 \end{aligned}$$

is on the singular curve $\tilde{E} : y^2 = x^3$.

In one of our attacks, the uniform selection of the secret resulted in

$$s = 109785787802901117004740481366937765753659896978.$$

That gives us the output $\tilde{Q} = (x_Q, y_Q) = s\tilde{P} \in \tilde{E}$ with

$$\begin{aligned} x_Q &= 405328711901172674829657351643194238623598390468 \\ y_Q &= 103085238587127999197783933077858660011787397349. \end{aligned}$$

Finally, we recover the secret s based on Theorem 2.11:

$$s_1 = \frac{\phi^+(\tilde{Q})}{\phi^+(\tilde{P})} = \frac{y_P}{x_P}\frac{x_Q}{y_Q} = \frac{\sqrt{x_U}x_Q}{y_Q} \mod p.$$

Because $r < p$, we only obtain one possible candidate with $s_1 = s$.

A.2. Singular curve attack for general j-invariant

In this section, we give a numerical example for our attack from Section 7.3. We use the curve `secp192r1` from [Cer10] that is defined over a field $\mathbb{F}_q$ of size $q = p = 2^{192} - 2^{64} - 1$. The curve is defined as $E : y^2 = x^3 - 3x + a_6$, with

$$a_6 = 2455155546008943817740293915197451784769108058161191238065$$

of prime order

$$r = 6277101735386680763835789423176059013767194773182842284081.$$

For the fault, we assume $\delta = -a_4 = 3$. This models an attack that completely skips the addition in Line 3 of Algorithm 3.1.

It turns out that for both choices $i = 0$ and $i = 1$, we obtain a square in step A-3. We choose $i = 0$ because in this case $x_S = (-1)^i = 1$ and

$$\alpha = \sqrt{3} = 232629722734768028008032747155364454195593722316376383590 2$$

is contained in $\mathbb{F}_q$. According to Theorem 2.11 this provides us with the more efficient reduction of the DLOG problem to $\mathbb{F}_q^*$. For $i = 0$ we obtain $\tilde{P} = (x_0, y_0)$ with

$$x_0 = 336634930825480590331042831040596034913290311420648622816 5$$
$$y_0 = 275212011998315130431081417702513312856431991660187772988 7.$$

This point is on the singular curve $\tilde{E} : y^2 = x^3 - 3x + 2$.

Random sampling of $\mathcal{B}$'s secret in $\mathbb{Z}/r\mathbb{Z}$ resulted in

$$s = 1113678563645930635777463018382920452293547545942927564878$$

and we obtain $\tilde{Q} = s\tilde{P} \in \tilde{E}$ with

$$x_Q = 4665052203107855484719784993687553637005598487381434479215$$
$$y_Q = 9288610891757447550076808419457331050787225599852136490 09.$$

Now, we apply Theorem 2.11. The images of $\tilde{P}$ and $\tilde{Q}$ under ϕ^* are given as

$$\phi^*(\tilde{P}) = 1544399762588312570436026623516397841564503871684736441528$$
$$\phi^*(\tilde{Q}) = 1050682542392422889161194344907633478108216148696830571786.$$

We used the function `znlog` of Pari/GP [PAR12] to compute s' as DLOG of $\phi^*(\tilde{Q})$ to the basis $\phi^*(\tilde{P})$. According to the documentation the implementation is based on the linear sieve index calculus method [COS86]. The computation took us 39 hours on one core of a 64 bit Intel Core i5 CPU with 8 GB of main memory.

Let d be the order of $\phi^*(\tilde{P})$. In this example we obtain a ratio of $r/d \approx 2$. Hence, $s \in \{s', s + d\}$ and in our particular example we find $s = s'$.

Acronyms

ABE	attribute-based encryption
AES	advanced encryption standard
BDH	bilinear Diffie-Hellman
BLS	Boneh-Lynn-Shacham
BN	Barreto-Naehrig
CAS	computer algebra system
CBDH	computational bilinear Diffie-Hellman
CCA	chosen ciphertext attack
CDH	computational Diffie-Hellman
CPA	chosen plaintext attack
CPU	central processing unit
CRT	Chinese remainder theorem
DDK	Die Datenkrake
DLOG	discrete logarithm
DPA	differential power analysis
ECC	elliptic curve cryptography
ECDSA	elliptic curve digital signature algorithm
ECIES	elliptic curve integrated encryption scheme
ECSM	elliptic curve scalar multiplication
EEA	extended Euclidean algorithm
EI	exponentiation inversion
EM	electromagnetic
FA	fault attack
FAPI	fixed argument pairing inversion
FPGA	field programmable gate array
HFP	hidden factor problem
HMP	hidden multiplier problem

HRP	hidden root problem
IBE	identity-based encryption
IND	indistinguishability
IPV	invalid point vulnerable
IPV1	first order IPV
IPV2	second order IPV
KDF	key derivation function
LSB	least significant bit
MI	Miller inversion
MSB	most significant bit
OpenSSL	Open Secure Socket Layer
PBC	pairing-based cryptography
RISC	reduced instruction set computer
RSA	Rivest-Shamir-Adleman
SCA	side-channel attack
TLS	Transport Layer Security

Notation

$\#S$	cardinality of the set S
$\oplus$	direct sum of groups
$\langle g \rangle$	subgroup generated by g
$[a, b]$	interval from a to b
$[n]$	multiplication-by-n map on an elliptic curve
$\lceil x \rceil_2^n$	zero-padded n-bit binary representation of x
Δ	discriminant of Weierstrass equation
μ_n	n-th roots of unity
$\Phi_n(x)$	n-th cyclotomic polynomial
π_q	q-th power Frobenius map
σ_q	q-th power Frobenius automorphism on $\overline{\mathbb{F}}_q$
$\mathbb{A}^n(\mathbb{K})$	n-dimensional affine space over $\mathbb{K}$
$\mathbb{A}^n$	abbreviation for $\mathbb{A}^n(\overline{\mathbb{K}})$
$a.b$	concatenation of a and b
$a \vert b$	a divides b
$a \parallel b$	a divides b exactly
ate_λ	reduced Ate pairing
$\deg(D)$	degree of divisor D
$\deg(f(x_1, \ldots, x_n))$	The total degree of $f(x_1, \ldots, x_n)$
$\text{div}(f)$	divisor of f
$E/\mathbb{K}$	elliptic curve E defined over field $\mathbb{K}$
$E[n]$	n-torsion points of E
$E(\mathbb{K})[n]$	$\mathbb{K}$-rational n-torsion points of E
$\mathbb{F}_q$	field with q elements
$f^q(x_1, \ldots, x_n)$	reduction of $f(x_1, \ldots, x_n)$ modulo σ_q
$\mathbb{G}_1(E, n, q)$	1-eigenspace of π_q on $E[n]$
$\mathbb{G}_2(E, n, q)$	q-eigenspace of π_q on $E[n]$
$\gcd(n_1, \ldots, n_k)$	greatest common divisor of $n_1, \ldots, n_k$

$\mathrm{HW}(a)$	Hamming weight of an integer a
j	j-invariant of Weierstrass equation
$\mathbb{K}^+$	additive subgroup of $\mathbb{K}$
$\mathbb{K}^*$	multiplicative subgroup of $\mathbb{K}$
$\overline{\mathbb{K}}$	algebraic closure of $\mathbb{K}$
$\mathbb{K}(\alpha_1, \ldots, \alpha_n)$	extension field of $\mathbb{K}$ generated by $\alpha_1, \ldots, \alpha_n$
$\mathbb{K}(x_1, \ldots, x_n)$	rational functions in $x_1, \ldots, x_n$ over $\mathbb{K}$
$\mathbb{K}[x_1, \ldots, x_n]$	polynomials in $x_1, \ldots, x_n$ over $\mathbb{K}$
$\mathbb{K}(X)$	function field of X over $\mathbb{K}$
$\ker(\phi)$	kernel of ϕ
$k(q, n)$	embedding degree of q and n
$\mathbb{L}/\mathbb{K}$	extension field $\mathbb{L}$ of $\mathbb{K}$
$\mathrm{mil}_{n,P}$	Miller function for n and P
$\mathrm{mon}(f(x_1, \ldots, x_n))$	monomials of $f(x_1, \ldots, x_n)$
$\mathcal{O}$	point at infinity of elliptic curve
$\mathrm{ord}(g)$	order of g
$\mathrm{ord}_P(f)$	order of f at P
$\mathbb{P}^n(\mathbb{K})$	n-dimensional projective space over $\mathbb{K}$
$\mathbb{P}^n$	abbreviation for $\mathbb{P}^n(\overline{\mathbb{K}})$
$\mathrm{sum}(D)$	sum of divisor D
$\mathrm{supp}(D)$	support of divisor D
t_n	Tate-Lichtenbaum pairing
$\hat{\mathrm{t}}_n$	reduced Tate-Lichtenbaum pairing
$V(\mathcal{F})$	affine algebraic set of $\mathcal{F} \subseteq \mathbb{K}[x_1, \ldots, x_n]$
v^T	transpose of v
$\mathrm{w}_q(a)$	weight of an integer a with respect to q
$X(\mathbb{K})$	$\mathbb{K}$-rational points of an algebraic set X

$x(P)$ x-coordinate of $P \in \mathbb{A}^2$

$y(P)$ y-coordinate of $P \in \mathbb{A}^2$

Bibliography

[Aca+13] Tolga Acar, Kristin E. Lauter, Michael Naehrig, and Daniel Shumow. "Affine Pairings on ARM." In: *Proceedings of Pairing-Based Cryptography — Pairing 2012*. Ed. by Michel Abdalla and Tanja Lange. Vol. 7708. LNCS. Springer, 2013, pp. 203–209.

[ADH12] Abdulaziz Alkhoraidly, Agustin Dominguez-Oviedo, and M. Anwar Hasan. "Fault Attacks on Elliptic Curve Cryptosystems." In: *Fault Analysis in Cryptography*. Ed. by Marc Joye and Michael Tunstall. Information Security and Cryptography. Springer, 2012, pp. 137–155.

[AG] Diego de Freitas Aranha and Conrado Porto Lopes Gouvêa. *RELIC is an Efficient LIbrary for Cryptography*. URL: `https://github.com/relic-toolkit/relic` (visited on 03/21/2014).

[Ago+10] Michel Agoyan, Jean-Max Dutertre, David Naccache, Bruno Robisson, and Assia Tria. "When Clocks Fail: On Critical Paths and Clock Faults." In: *Proceedings of Smart Card Research and Advanced Application (CARDIS) 2010*. Ed. by Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny. Vol. 6035. LNCS. Springer, 2010, pp. 182–193.

[AK96] Ross Anderson and Markus Kuhn. "Tamper Resistance: A Cautionary Note." In: *Proceedings of USENIX Workshop on Electronic Commerce 1996*. USENIX. USENIX Association, 1996, pp. 1–11.

[ANS99] ANSI X9 Working Group. *ANSI Standard Public Key Cryptography for the Financial Service Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). ANSI Std X9.62-1998*. American National Standards Institute (ANSI), 1999.

[Ant+03] Adrian Antipa, Daniel R. L. Brown, Alfred Menezes, René Struik, and Scott A. Vanstone. "Validation of Elliptic Curve Public Keys." In: *Proceedings of Public Key Cryptography (PKC) 2003*. Ed. by Yvo Desmedt. Vol. 2567. LNCS. Springer, 2003, pp. 211–223.

[Atm12] Atmel Corporation. *XMEGA A Manual*. 2012. URL: `www.atmel.com`.

[Atm13] Atmel Corporation. *ATxmega128A1/ATxmega64A1 Datasheet*. 2013. URL: `www.atmel.com`.

[Atm14] Atmel Corporation. *Atmel AVR 8-bit Instruction Set*. 2014. URL: `www.atmel.com`.

[Aum+03] Christian Aumüller, Peter Bier, Wieland Fischer, Peter Hofreiter, and Jean-Pierre Seifert. "Fault Attacks on RSA with CRT: Concrete Results and Practical Countermeasures." In: *Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2002*. Ed. by Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar. Vol. 2523. LNCS. Springer, 2003, pp. 260–275.

[Bar+06] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. "The Sorcerer's Apprentice Guide to Fault Attacks." In: *Proceedings of the IEEE* 94.2 (2006), pp. 370–382.

[Bar+07] Paulo S. L. M. Barreto, Steven D. Galbraith, Colm O'Eigeartaigh, and Michael Scott. "Efficient Pairing Computation on supersingular Abelian Varieties." In: *Designes, Codes and Cryptography* 42.3 (2007), pp. 239–271.

[Bar+12a] Alessandro Barenghi, Guido M. Bertoni, Luca Breveglieri, Mauro Pelliccioli, and Gerardo Pelosi. "Injection Technologies for Fault Attacks on Microprocessors." In: *Fault Analysis in Cryptography*. Ed. by Marc Joye and Michael Tunstall. Information Security and Cryptography. Springer, 2012, pp. 275–293.

[Bar+12b] Alessandro Barenghi, Luca Breveglieri, Israel Koren, and David Naccache. "Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures." In: *Proceedings of the IEEE* 100.11 (2012), pp. 3056–3076.

[BB05] David Brumley and Dan Boneh. "Remote timing Attacks are Practical." In: *Computer Networks* 48.5 (2005), pp. 701–716.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. "On the Importance of Checking Cryptographic Protocols for Faults." In: *Proceedings of Advances in Cryptology (EUROCRYPT) 1997*. Ed. by Walter Fumy. Vol. 1233. LNCS. Springer, 1997, pp. 37–51.

[Bel+15] Sonia Belaïd, Jean-Sébastien Coron, Pierre-Alain Fouque, Benoît Gérard, Jean-Gabriel Kammerer, and Emmanuel Prouff. "Improved Side-Channel Analysis of Finite-Field Multiplication." In: *Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2015*. Ed. by Tim Güneysu and Helena Handschuh. Vol. 9293. LNCS. Springer, 2015, pp. 395–415.

[Beu+10] Jean-Luc Beuchat, Jorge E. González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. "High-Speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves." In: *Proceedings of Pairing-Based Cryptography — Pairing 2010*. Ed. by Marc Joye, Atsuko Miyaji, and Akira Otsuka. Vol. 6487. LNCS. Springer, 2010, pp. 21–39.

[BF01] Dan Boneh and Matthew K. Franklin. "Identity-Based Encryption from the Weil Pairing." In: *Proceedings of Advances in Cryptology (CRYPTO) 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, 2001, pp. 213–229.

[BF03] Dan Boneh and Matthew K. Franklin. "Identity-Based Encryption from the Weil Pairing." In: *SIAM Journal on Computing* 32.3 (2003), pp. 586–615.

[BG15] Johannes Blömer and Peter Günther. "Singular Curve Point Decompression Attack." In: *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2015*. Ed. by Naofumi Homma and Victor Lomné. IEEE Computer Society, 2015, pp. 71–84.

[BGK13] Johannes Blömer, Peter Günther, and Volker Krummel. "Securing Critical Unattended Systems with Identity Based Cryptography — A Case Study." In: *Proceedings of Mathematical Aspects of Computer and Information Sciences (MACIS) 2013*. 2013, pp. 98–105.

[BGL11] Johannes Blömer, Peter Günther, and Gennadij Liske. "Improved Side Channel Attacks on Pairing Based Cryptography." In: *IACR Cryptology ePrint Archive* (2011). https://eprint.iacr.org: `ia.cr/2011/706` (2011/706).

[BGL13] Johannes Blömer, Peter Günther, and Gennadij Liske. "Improved Side Channel Attacks on Pairing Based Cryptography." In: *Proceedings of Constructive Side-Channel Analysis and Secure Design (COSADE) 2013*. Ed. by Emmanuel Prouff. Vol. 7864. LNCS. Springer, 2013, pp. 154–168.

[BGL14] Johannes Blömer, Peter Günther, and Gennadij Liske. "Tampering Attacks in Pairing-Based Cryptography." In: *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2014*. Ed. by Assia Tria and Dooho Choi. IEEE Computer Society, 2014, pp. 1–7.

[BGV11] Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. "An In-depth and Black-box Characterization of the Effects of Clock Glitches on 8-bit MCUs." In: *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2011*. Ed. by Luca Breveglieri, Sylvain Guilley, Israel Koren, David Naccache, and Junko Takahashi. IEEE Computer Society, 2011, pp. 105–114.

[Bit15] Bitcoin Project. *Bitcoin developer documentation*. 2015. URL: `https://bitcoin.org/en/developer-documentation` (visited on 10/27/2015).

[BK98] R. Balasubramanian and Neal Koblitz. "The Improbability that an Elliptic Curve has Subexponential Discrete Log Problem under the Menezes-Okamoto-Vanstone Algorithm." In: *Journal of Cryptology* 11.2 (1998), pp. 141–145.

[Blö+14] Johannes Blömer, Ricardo Gomes da Silva, Peter Günther, Juliane Krämer, and Jean-Pierre Seifert. "A Practical Second-Order Fault Attack against a Real-World Pairing Implementation." In: *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2014*. Ed. by Assia Tria and Dooho Choi. IEEE Computer Society, 2014, pp. 123–136.

[BLS04a] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. "On the Selection of Pairing-Friendly Groups." In: *Proceedings of Selected Areas in Cryptography (SAC) 2003*. Ed. by Mitsuru Matsui and Robert J. Zuccherato. Vol. 3006. LNCS. Springer, 2004, pp. 17–25.

[BLS04b] Dan Boneh, Ben Lynn, and Hovav Shacham. "Short Signatures from the Weil Pairing." In: *Journal of Cryptology* 17.4 (2004), pp. 297–319.

[BM07] Xavier Boyen and Luther Martin. *Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems (RFC 5091)*. Internet Engineering Task Force. 2007. URL: `http://www.ietf.org/rfc/rfc5091.txt`.

[BMH13] Kiseok Bae, Sangjae Moon, and Jaecheol Ha. "Instruction Fault Attack on the Miller Algorithm in a Pairing-Based Cryptosystem." In: *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2013*. Ed. by Leonard Barolli, Ilsun You, Fatos Xhafa, Fang-Yie Leu, and Hsing-Chung Chen. IEEE Computer Society, 2013, pp. 167–174.

[BMM00] Ingrid Biehl, Bernd Meyer, and Volker Müller. "Differential Fault Attacks on Elliptic Curve Cryptosystems." In: *Proceedings of Advances in Cryptology (CRYPTO) 2000*. Ed. by Mihir Bellare. Vol. 1880. LNCS. Springer, 2000, pp. 131–146.

[BN06] Paulo S. L. M. Barreto and Michael Naehrig. "Pairing-Friendly Elliptic Curves of Prime Order." In: *Proceedings of Selected Areas in Cryptography (SAC) 2005*. Ed. by Bart Preneel and Stafford E. Tavares. Vol. 3897. LNCS. Springer, 2006.

[Bol03] Alexandra Boldyreva. "Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme." In: *Proceedings of Public Key Cryptography (PKC) 2003*. Ed. by Yvo Desmedt. Vol. 2567. LNCS. Springer, 2003, pp. 31–46.

[Bon+03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps." In: *Proceedings of Advances in Cryptology (EUROCRYPT) 2003*. Ed. by Eli Biham. Vol. 2656. LNCS. Springer, 2003, pp. 416–432.

[BOS06] Johannes Blömer, Martin Otto, and Jean-Pierre Seifert. "Sign Change Fault Attacks on Elliptic Curve Cryptosystems." In: *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2006*. Ed. by Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert. Vol. 4236. LNCS. Springer, 2006, pp. 36–52.

[BRD01] Bundesrepublik Deutschland. "Gesetz über Rahmenbedingungen für elektronische Signaturen und zur Änderung weiterer Vorschriften." German. In: *Bundesgesetzblatt Teil I* 22 (2001), pp. 867–884.

[Bre+08] Luca Breveglieri, Shay Gueron, Israel Koren, David Naccache, and Jean-Pierre Seifert, eds. *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2008*. IEEE Computer Society, 2008.

[BS04] Dan Boneh and Hovav Shacham. "Group signatures with verifier-local revocation." In: *Proceedings of Conference on Computer and Communications Security (CCS) 2004*. Ed. by Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick Drew McDaniel. Association for Computing Machinery (ACM), 2004, pp. 168–177.

[BS97] Eli Biham and Adi Shamir. "Differential Fault Analysis of Secret Key Cryptosystems." In: *Proceedings of Advances in Cryptology (CRYPTO) 1997*. Ed. by Burton S. Kaliski Jr. Vol. 1294. LNCS. Springer, 1997, pp. 513–525.

[BSS05] Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart, eds. *Advances in Elliptic Curve Cryptography*. London Mathematical Society Lecture Note Series 317. Cambridge University Press, 2005.

[CC03] Jae Choon Cha and Jung Hee Cheon. "An Identity-Based Signature from Gap Diffie-Hellman Groups." In: *Proceedings of Public Key Cryptography (PKC) 2003*. Ed. by Yvo Desmedt. Vol. 2567. LNCS. Springer, 2003, pp. 18–30.

[CCS07] Liqun Chen, Zhaohui Cheng, and Nigel P. Smart. "Identity-Based Key Agreement Protocols from Pairings." In: *International Journal of Information Security* 6.4 (2007), pp. 213–241.

[Cer10] Certicom Research. *Standards for Efficient Cryptography 2 (SEC 2): Recommended Elliptic Curve Domain Parameters*. Standards for Efficient Cryptography Group (SECG), 2010.

[Cha+10] Sanjit Chatterjee, Darrel Hankerson, Edward Knapp, and Alfred Menezes. "Comparing two Pairing-Based Aggregate Signature Schemes." In: *Designs, Codes and Cryptography* 55.2-3 (2010), pp. 141–167.

[Cha+14] Seunghwan Chang, Hoon Hong, Eunjeong Lee, and Hyang-Sook Lee. "Pairing Inversion via Non-degenerate Auxiliary Pairings." In: *Proceedings of Pairing-Based Cryptography — Pairing 2013*. Ed. by Zhenfu Cao and Fangguo Zhang. Vol. 8365. LNCS. Springer, 2014, pp. 77–96.

[CHK08] Dooho Choi, Dongguk Han, and Howon Kim. "Construction of Efficient and Secure Pairing Algorithm and its Application." In: *Journal of Communications and Networks* 10.4 (2008), pp. 437–443.

[CHM10] Sanjit Chatterjee, Darrel Hankerson, and Alfred Menezes. "On the Efficiency and Security of Pairing-Based Protocols in the Type 1 and Type 4 Settings." In: *Proceedings of Arithmetic of Finite Fields (WAIFI) 2010*. Ed. by M. Anwar Hasan and Tor Helleseth. Vol. 6087. LNCS. Springer, 2010, pp. 114–134.

[CJ05] Mathieu Ciet and Marc Joye. "Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults." In: *Designs, Codes and Cryptography* 36.1 (2005), pp. 33–43.

[CKM15] Sanjit Chatterjee, Koray Karabina, and Alfred Menezes. "Fault Attacks on Pairing-Based Protocols Revisited." In: *IEEE Transactions on Computers* 64.6 (2015), pp. 1707–1714.

[CLN10] Craig Costello, Tanja Lange, and Michael Naehrig. "Faster Pairing Computations on Curves with High-Degree Twists." In: *Proceedings of Public Key Cryptography (PKC) 2010*. Ed. by Phong Q. Nguyen and David Pointcheval. Vol. 6056. LNCS. Springer, 2010, pp. 224–242.

[CLO96] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra*. 2nd ed. Undergraduate Texts in Mathematics. Springer, 1996.

[CM11] Sanjit Chatterjee and Alfred Menezes. "On Cryptographic Protocols Employing Asymmetric Pairings — The Role of Ψ Revisited." In: *Discrete Applied Mathematics* 159.13 (2011), pp. 1311–1322.

[Coh+06] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Discrete Mathematics and its Applications. Chapman & Hall/CRC, 2006.

[Cor99] Jean-Sébastien Coron. "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems." In: *Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 1999*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1717. LNCS. Springer, 1999, pp. 292–302.

[COS86] Don Coppersmith, Andrew M. Odlyzko, and Richard Schroeppel. "Discrete Logarithms in GF(p)." In: *Algorithmica* 1.1 (1986), pp. 1–15.

[CT05] Hamid Choukri and Michael Tunstall. "Round Reduction Using Faults." In: *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2005*. Ed. by Luca Breveglieri and Israel Koren. LNCS. Springer, 2005, pp. 13–24.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. "Unifying Leakage Models: From Probing Attacks to Noisy Leakage." In: *Proceedings of Advances in Cryptology (EUROCRYPT) 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, 2014, pp. 423–440.

[Des03] Yvo Desmedt, ed. *Proceedings of Public Key Cryptography (PKC) 2003*. Vol. 2567. LNCS. Springer, 2003.

[DH11] Agustin Dominguez-Oviedo and M. Anwar Hasan. "Algorithm-level Error Detection for Montgomery Ladder-based ECSM." In: *Journal of Cryptographic Engineering* 1.1 (2011), pp. 57–69.

[DH76] Whitfield Diffie and Martin E. Hellman. "New Directions in Cryptography." In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654.

[DL03] Iwan M. Duursma and Hyang-Sook Lee. "Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$." In: *Proceedings of Advances in Cryptology (ASIACRYPT) 2013*. Ed. by Chi-Sung Laih. Vol. 2894. LNCS. Springer, 2003, pp. 111–123.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. "Leakage-Resilient Cryptography." In: *Proceedings of Foundations of Computer Science (FOCS) 2008*. IEEE Computer Society, 2008, pp. 293–302.

[FH08] Julie Ferrigno and Martin Hlavác. "When AES Blinks: Introducing Optical Side Channel." In: *IET Information Security* 2.3 (2008), pp. 94–98.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes." In: *Journal of Cryptology* 26.1 (2013), pp. 80–101.

[Fou+08a] Pierre-Alain Fouque, Reynald Lercier, Denis Réal, and Frédéric Valette. "Fault Attack on Elliptic Curve Montgomery Ladder Implementation." In: *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2008*. Ed. by Luca Breveglieri, Shay Gueron, Israel Koren, David Naccache, and Jean-Pierre Seifert. IEEE Computer Society, 2008, pp. 92–98.

[Fou+08b] Pierre-Alain Fouque, Denis Réal, Frédéric Valette, and M'hamed Drissi. "The Carry Leakage on the Randomized Exponent Countermeasure." In: *Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2008*. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Vol. 5154. LNCS. Springer, 2008, pp. 198–213.

[FST10] David Freeman, Michael Scott, and Edlyn Teske. "A Taxonomy of Pairing-Friendly Elliptic Curves." In: *Journal of Cryptology* 23.2 (2 2010), pp. 224–280.

[FV12] Junfeng Fan and Ingrid Verbauwhede. "An Updated Survey on Secure ECC Implementations: Attacks, Countermeasures and Cost." 2012, pp. 265–282.

[Gal12] Steven D. Galbraith. *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012.

[Gal13] Steven Galbraith. *Recent Work on Pairing Inversion*. Blog. Nov. 6, 2013. URL: `https://ellipticnews.wordpress.com` (visited on 05/26/2015).

[GC11] Santosh Ghosh and Dipanwita Roy Chowdhury. "Security of Prime Field Pairing Cryptoprocessor against Differential Power Attack." In: *Proceedings of Security Aspects in Information Technology 2011*. Ed. by Marc Joye, Debdeep Mukhopadhyay, and Michael Tunstall. Vol. 7011. LNCS. Springer, 2011, pp. 16–29.

[GCC15] GCC team. *avr-gcc Wiki*. Free Software Foundation. 2015. URL: `https://gcc.gnu.org/wiki/avr-gcc` (visited on 10/06/2015).

[GhS07] Steven D. Galbraith, Colm O. hEigeartaigh, and Caroline Sheedy. "Simplified Pairing Computation and Security Implications." In: *Journal of Mathematical Cryptology* 1.3 (2007), pp. 267–281.

[GHV08] Steven D. Galbraith, Florian Hess, and Frederik Vercauteren. "Aspects of Pairing Inversion." In: *IEEE Transactions on Information Theory* 54.12 (2008), pp. 5719–5728.

[GK15] Peter Günther and Volker Krummel. "Implementing Cryptographic Pairings on Accumulator based Smart Card Architectures." In: *Proceedings of Mathematical Aspects of Computer and Information Sciences (MACIS) 2015*. Ed. by Ilias S. Kotsireas, Siegfried M. Rump, and Chee K. Yap. Vol. 9582. LNCS. Springer, 2015, pp. 151–165.

[GLV01] Robert P. Gallant, Robert J. Lambert, and Scott A. Vanstone. "Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms." In: *Proceedings of Advances in Cryptology (CRYPTO) 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, 2001, pp. 190–200.

[GMM15] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. "Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript." In: (2015). arXiv: `1507.06955 [cs.CR]`. URL: `http://arxiv.org/abs/1507.06955`.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. "Electromagnetic Analysis: Concrete Results." In: *Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2001*. Ed. by Çetin Kaya Koç, David Naccache, and Christof Paar. Vol. 2162. LNCS. Springer, 2001, pp. 251–261.

[Gom14] Ricardo Gomes da Silva. "Practical Analysis of Embedded Microcontrollers against Clock Glitching Attacks." Bachelor's thesis. Technische Universität Berlin, 2014. URL: `http://rgsilva.com/Bachelorarbeit.pdf` (visited on 08/27/2015).

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Perfect Non-interactive Zero Knowledge for NP." In: *Proceedings of Advances in Cryptology (EUROCRYPT) 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, 2006, pp. 339–358.

[GP08] Steven D. Galbraith and Kenneth G. Paterson, eds. *Proceedings of Pairing-Based Cryptography — Pairing 2008*. Vol. 5209. LNCS. Springer, 2008.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. "Pairings for Cryptographers." In: *Discrete Applied Mathematics* 156.16 (2008), pp. 3113–3121.

[GS08] Steven D. Galbraith and Michael Scott. "Exponentiation in Pairing-Friendly Groups Using Homomorphisms." In: *Proceedings of Pairing-Based Cryptography — Pairing 2008*. Ed. by Steven D. Galbraith and Kenneth G. Paterson. Vol. 5209. LNCS. Springer, 2008, pp. 211–224.

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. "RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis." In: *Proceedings of Advances in Cryptology (CRYPTO) 2014*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, 2014, pp. 444–461.

[HS07] Michael Hutter and Jörn-Marc Schmidt. "Optical and EM fault-attacks on CRT-based RSA: Concrete results." In: *Proceedings of Austrochip 2007*. Verlag der Technischen Universität Graz, 2007, pp. 61–67.

[HSV06] Florian Hess, Nigel P. Smart, and Frederik Vercauteren. "The Eta Pairing Revisited." In: *IEEE Transactions on Information Theory* 52.10 (2006), pp. 4595–4602.

[Hut+09] Michael Hutter, Marcel Medwed, Daniel M. Hein, and Johannes Wolkerstorfer. "Attacking ECDSA-Enabled RFID Devices." In: *Proceedings of Applied Cryptography and Network Security (ACNS) 2009*. Ed. by Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud. Vol. 5536. LNCS. Springer, 2009, pp. 519–534.

[IEE00] IEEE P1363 Working Group. *IEEE Standard Specifications for Public-Key Cryptography. IEEE Std 1363-2000*. IEEE Computer Society, 2000.

[IEE13] IEEE P1363 Working Group. *IEEE Standard for Identity-Based Cryptographic Techniques using Pairings. IEEE Std 1363.3-2013*. IEEE Computer Society, 2013.

[JKP03] Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, eds. *Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2002*. Vol. 2523. LNCS. Springer, 2003.

[JN09] Marc Joye and Gregory Neven, eds. *Identity-Based Cryptography*. Vol. 2. Cryptology and Information Security. IOS Press, 2009.

[Jou+06] Antoine Joux, Reynald Lercier, Nigel Smart, and Frederik Vercauteren. "The Number Field Sieve in the Medium Prime Case." In: *Proceedings of Advances in Cryptology (CRYPTO) 2006*. Ed. by Cynthia Dwork. Vol. 4117. LNCS. Springer, 2006, pp. 326–344.

[Jou04] Antoine Joux. "A One Round Protocol for Tripartite Diffie-Hellman." In: *Journal of Cryptology* 17.4 (2004), pp. 263–276.

[JT12] Marc Joye and Michael Tunstall, eds. *Fault Analysis in Cryptography*. Information Security and Cryptography. Springer, 2012.

[Kil01] Joe Kilian, ed. *Proceedings of Advances in Cryptology (CRYPTO) 2001*. Vol. 2139. LNCS. Springer, 2001.

[Kim+06] Tae Hyun Kim, Tsuyoshi Takagi, Dong-Guk Han, Ho Won Kim, and Jongin Lim. "Side Channel Attacks and Countermeasures on Pairing Based Cryptosystems over Binary Fields." In: *Proceedings of Cryptology and Network Security (CANS) 2006*. Ed. by David Pointcheval, Yi Mu, and Kefei Chen. Vol. 4301. LNCS. Springer, 2006, pp. 168–181.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis." In: *Proceedings of Advances in Cryptology (CRYPTO) 1999*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, 1999, pp. 388–397.

[KL08] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. 1st ed. Chapman & Hall/CRC, 2008.

[KM13] Neal Koblitz and Alfred Menezes. "Another Look at Security Definitions." In: *Advances in Mathematics of Communications* 7.1 (2013), pp. 1–38.

[KO12] Naoki Kanayama and Eiji Okamoto. "Approach to Pairing Inversions Without Solving Miller Inversion." In: *IEEE Transactions on Information Theory* 58.2 (2012), pp. 1248–1253.

[Koc96] Paul C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems." In: *Proceedings of Advances in Cryptology (CRYPTO) 1996*. Ed. by Neal Koblitz. Vol. 1109. LNCS. Springer, 1996, pp. 104–113.

[KP06] Bo Gyeong Kang and Je Hong Park. "On the Relationship between Squared Pairings and Plain Pairings." In: *Information Processing Letters* 97.6 (2006), pp. 219–224.

[KQ07] Chong Hee Kim and Jean-Jacques Quisquater. "Fault Attacks for CRT Based RSA: New Attacks, New Results, and New Countermeasures." In: *Proceedings of Information Security Theory and Practices (WISTP) 2007*. Ed. by Damien Sauveron, Constantinos Markantonakis, Angelos Bilas, and Jean-Jacques Quisquater. Vol. 4462. LNCS. Springer, 2007, pp. 215–228.

[KSS08] Ezekiel J. Kachisa, Edward F. Schaefer, and Michael Scott. "Constructing Brezing-Weng Pairing-Friendly Elliptic Curves Using Elements in the Cyclotomic Field." In: *Proceedings of Pairing-Based Cryptography — Pairing 2008*. Ed. by Steven D. Galbraith and Kenneth G. Paterson. Vol. 5209. LNCS. Springer, 2008, pp. 126–135.

[KSV13] Dusko Karaklajic, Jörn-Marc Schmidt, and Ingrid Verbauwhede. "Hardware Designer's Guide to Fault Attacks." In: *IEEE Transactions on VLSI Systems* 21.12 (2013), pp. 2295–2306.

[KU10] Koray Karabina and Berkant Ustaoglu. "Invalid-Curve Attacks on (Hyper)elliptic Curve Cryptosystems." In: *Advances in Mathematics of Communications* 3 (2010), pp. 307–321.

[Lan02] Serge Lang. *Algebra*. 3rd ed. Graduate Texts in Mathematics 211. Springer, 2002.

[Las+14] Ronan Lashermes, Marie Paindavoine, Nadia El Mrabet, Jacques J. A. Fournier, and Louis Goubin. "Practical Validation of Several Fault Attacks against the Miller Algorithm." In: *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2014*. Ed. by Assia Tria and Dooho Choi. IEEE Computer Society, 2014, pp. 115–122.

[LFG13] Ronan Lashermes, Jacques Fournier, and Louis Goubin. "Inverting the Final Exponentiation of Tate Pairings on Ordinary Elliptic Curves using Faults." In: *Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2013*. Ed. by Guido Bertoni and Jean-Sébastien Coron. Vol. 8086. LNCS. Springer, 2013, pp. 365–382.

[LLP09] E. Lee, H.-S. Lee, and C.-M. Park. "Efficient and Generalized Pairing Computation on Abelian Varieties." In: *IEEE Transactions on Information Theory* 55.4 (2009), pp. 1793–1803.

[LQ04] Benoît Libert and Jean-Jacques Quisquater. "Efficient Signcryption with Key Privacy from Gap Diffie-Hellman Groups." In: *Proceedings of Public Key Cryptography (PKC) 2004*. Ed. by Feng Bao, Robert H. Deng, and Jianying Zhou. Vol. 2947. LNCS. Springer, 2004, pp. 187–200.

[Mes00] Thomas S. Messerges. "Using Second-Order Power Analysis to Attack DPA Resistant Software." In: *Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2000*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1965. LNCS. Springer, 2000, pp. 238–251.

[MF15] Nadia El Mrabet and Emmanuel Fouotsa. "Failure of the Point Blinding Countermeasure Against Fault Attack in Pairing-Based Cryptography." In: *Proceedings of Codes, Cryptology, and Information Security (C2SI) 2015*. Ed. by Said El Hajji, Abderrahmane Nitaj, Claude Carlet, and El Mamoun Souidi. Vol. 9084. LNCS. Springer, 2015, pp. 259–273.

[MFN09] Nadia El Mrabet, Marie Lise Flottes, and Giorgio Di Natale. "A practical Differential Power Analysis attack against the Miller algorithm." In: *Conference on Ph. D. Research in Microelectronics and Electronics*. 2009, pp. 308–311.

[Mil04] Victor S. Miller. "The Weil Pairing, and its Efficient Calculation." In: *Journal of Cryptology* 17.4 (2004), pp. 235–261.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks. Revealing the Secrets of Smart Cards*. Springer, 2007.

[MOV93] Alfred J. Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. "Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field." In: *IEEE Transactions on Information Theory* 39.5 (1993), pp. 1639–1646.

[MPV12] Nadia El Mrabet, Dan Page, and Frederik Vercauteren. "Fault Attacks on Pairing-Based Cryptography." In: *Fault Analysis in Cryptography*. Ed. by Marc Joye and Michael Tunstall. Information Security and Cryptography. Springer, 2012, pp. 221–236.

[MR04] Silvio Micali and Leonid Reyzin. "Physically Observable Cryptography." In: *Proceedings of Theory of Cryptography Conference (TCC) 2004*. Ed. by Moni Naor. Vol. 2951. LNCS. Springer, 2004, pp. 278–296.

[Mra+15] Nadia El Mrabet, Jacques J. A. Fournier, Louis Goubin, and Ronan Lashermes. "A Survey of Fault Attacks in Pairing Based Cryptography." In: *Cryptography and Communications* 7.1 (2015), pp. 185–205.

[Mra09] Nadia El Mrabet. "What about Vulnerability to a Fault Attack of the Miller's Algorithm During an Identity Based Protocol?" In: *Proceedings of Information Security and Assurance (ISA) 2009*. Ed. by Jong Hyuk Park, Hsiao-Hwa Chen, Mohammed Atiquzzaman, Changhoon Lee, Tai-Hoon Kim, and Sang-Soo Yeo. Vol. 5576. LNCS. Springer, 2009, pp. 122–134.

[Mra10] Nadia El Mrabet. "Fault Attacks against the Miller's Algorithm in Edwards Coordinates." In: *Proceedings of Information Security and Assurance (ISA) 2010*. Ed. by Samir Kumar Bandyopadhyay, Wael Adi, Tai-Hoon Kim, and Yang Xiao. Vol. 76. Communications in Computer and Information Science. Springer, 2010, pp. 72–85.

[Nae09] Michael Naehrig. "Constructive and Computational Aspects of Cryptographic Pairings." PhD thesis. Eindhoven University of Technology, 2009.

[NS13] Dmitry Nedospasov and Thorsten Schröder. "Introducing Die Datenkrake: Programmable Logic for Hardware Security Analysis." In: *Proceedings of Workshop on Offensive Technologies (WOOT) 2013*. Ed. by Jon Oberheide and William K. Robertson. USENIX Association, 2013.

[Oli+11] Leonardo B. Oliveira, Diego F. Aranha, Conrado Porto Lopes Gouvêa, Michael Scott, Danilo F. Câmara, Julio López, and Ricardo Dahab. "TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks." In: *Computer Communications* 34.3 (2011), pp. 485–493.

[PAR12] PARI Group. *PARI/GP*. Version 2.5.1. 2012. URL: `http://pari.math.u-bordeaux.fr/`.

[PM11] Weibo Pan and William P. Marnane. "A Correlation Power Analysis Attack against Tate Pairing on FPGA." In: *Proceedings of Reconfigurable Computing: Architectures, Tools and Applications (ARC) 2011*. Ed. by Andreas Koch, Ram Krishnamurthy, John McAllister, Roger F. Woods, and Tarek A. El-Ghazawi. Vol. 6578. LNCS. Springer, 2011, pp. 340–349.

[PSM11] Jea Hoon Park, Gyo Yong Sohn, and Sang-Jae Moon. "Fault Attack on a Point Blinding Countermeasure of Pairing Algorithms." In: *ETRI Journal* 33.6 (2011), pp. 989–992.

[PV04] Dan Page and Frederik Vercauteren. "Fault and Side-Channel Attacks on Pairing Based Cryptography." In: *IACR Cryptology ePrint Archive* (2004). https://eprint.iacr.org: `ia.cr/2004/283` (2004/283).

[PV06] Dan Page and Frederik Vercauteren. "A Fault Attack on Pairing-Based Cryptography." In: *IEEE Transactions on Computers* 55.9 (2006), pp. 1075–1080.

[Pyt14] Python Software Foundation. *Python Programming Language Homepage*. 2014. URL: `http://www.python.org` (visited on 05/15/2014).

[QS01] Jean-Jacques Quisquater and David Samyde. "ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards." In: *Proceedings of Smart Card Programming and Security (E-smart) 2001*. Ed. by Isabelle Attali and Thomas P. Jensen. Vol. 2140. LNCS. Springer, 2001, pp. 200–210.

[Riv+15] Lionel Rivière, Zakaria Najm, Pablo Rauzy, Jean-Luc Danger, Julien Bringer, and Laurent Sauvage. "High Precision Fault Injections on the Instruction Cache of ARMv7-M Architectures." In: *Proceedings of Hardware-Oriented Security and Trust (HOST) 2015*. IEEE Computer Society, 2015, pp. 62–67.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[RW13] Yannis Rouselakis and Brent Waters. "Practical Constructions and New Proof Methods for Large Universe Attribute-Based Encryption." In: *Proceedings of Conference on Computer and Communications Security (CCS) 2013*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. Association for Computing Machinery (ACM), 2013, pp. 463–474.

[SA02] Sergei P. Skorobogatov and Ross J. Anderson. "Optical Fault Induction Attacks." In: *Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2002*. Ed. by Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar. Vol. 2523. LNCS. Springer, 2002, pp. 2–12.

[Sag14] Sage Developers. *SageMath Homepage*. Version 6.1. The SAGE Foundation. 2014. URL: `http://www.sagemath.org` (visited on 12/13/2015).

[Sat07] Takakazu Satoh. "On Pairing Inversion Problems." In: *Proceedings of Pairing-Based Cryptography — Pairing 2007*. Ed. by Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto. Vol. 4575. LNCS. Springer, 2007, pp. 317–328.

[Sat13] Takakazu Satoh. "On a Relation between the Ate Pairing and the Weil Pairing for Supersingular Elliptic Curves." In: *IACR Cryptology ePrint Archive* (2013). https://eprint.iacr.org: `ia.cr/2013/532` (2013/532).

[SCA06] Michael Scott, Neil Costigan, and Wesam Abdulwahab. "Implementing Cryptographic Pairings on Smartcards." In: *Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2006*. Ed. by Louis Goubin and Mitsuru Matsui. Vol. 4249. LNCS. Springer, 2006, pp. 134–147.

[Sch+13] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. "Simple Photonic Emission Analysis of AES." In: *Journal of Cryptographic Engineering* 3.1 (2013), pp. 3–15.

[Sco+09] Michael Scott, Naomi Benger, Manuel Charlemagne, Luis J. Dominguez Perez, and Ezekiel J. Kachisa. "On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves." In: *Proceedings of Pairing-Based Cryptography — Pairing 2009*. Ed. by Hovav Shacham and Brent Waters. Vol. 5671. LNCS. Springer, 2009, pp. 78–88.

[SH08] Jörn-Marc Schmidt and Christoph Herbst. "A Practical Fault Attack on Square and Multiply." In: *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2008*. Ed. by Luca Breveglieri, Shay Gueron, Israel Koren, David Naccache, and Jean-Pierre Seifert. IEEE Computer Society, 2008, pp. 53–58.

[Sha85] Adi Shamir. "Identity-Based Cryptosystems and Signature Schemes." In: *Proceedings of Advances in Cryptology (CRYPTO) 1984*. Ed. by G. R. Blakley and David Chaum. Vol. 196. LNCS. Springer, 1985, pp. 47–53.

[Sil09] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*. 2nd ed. Vol. 106. Graduate Texts in Mathematics. Springer, 2009.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks." In: *Proceedings of Advances in Cryptology (EUROCRYPT) 2009*. Ed. by Antoine Joux. Vol. 5479. LNCS. Springer, 2009, pp. 443–461.

[STO08] Masaaki Shirase, Tsuyoshi Takagi, and Eiji Okamoto. "An Efficient Countermeasure against Side Channel Attacks for Pairing Computation." In: *Proceedings of Information Security Practice and Experience (ISPEC) 2008*. Ed. by Liqun Chen, Yi Mu, and Willy Susilo. Vol. 4991. LNCS. Springer, 2008, pp. 290–303.

[SW05] Amit Sahai and Brent Waters. "Fuzzy Identity-Based Encryption." In: *Proceedings of Advances in Cryptology (EUROCRYPT) 2005*. Ed. by Ronald Cramer. Vol. 3494. LNCS. Springer, 2005, pp. 457–473.

[Tak+07] Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, eds. *Proceedings of Pairing-Based Cryptography — Pairing 2007*. Vol. 4575. LNCS. Springer, 2007.

[TC14] Assia Tria and Dooho Choi, eds. *Proceedings of Fault Diagnosis and Tolerance in Cryptography (FDTC) 2014*. IEEE Computer Society, 2014.

[UW14] Thomas Unterluggauer and Erich Wenger. "Practical Attack on Bilinear Pairings to Disclose the Secrets of Embedded Devices." In: *Proceedings of Availability, Reliability and Security (ARES) 2014*. IEEE Computer Society, 2014, pp. 69–77.

[Ver08] Frederik Vercauteren. "The Hidden Root Problem." In: *Proceedings of Pairing-Based Cryptography — Pairing 2008*. Ed. by Steven D. Galbraith and Kenneth G. Paterson. Vol. 5209. LNCS. Springer, 2008, pp. 89–99.

[Ver09] Frederic Vercauteren. "Pairings on Elliptic Curves." In: *Identity-Based Cryptography*. Ed. by Marc Joye and Gregory Neven. Vol. 2. Cryptology and Information Security. IOS Press, 2009, pp. 13–30.

[Ver10] Frederik Vercauteren. "Optimal pairings." In: *IEEE Transactions on Information Theory* 56.1 (2010), pp. 455–461.

[Wag12] Mathias Wagner. "700+ Attacks Published on Smart Cards: The Need for a Systematic Counter Strategy." In: *Proceedings of Constructive Side-Channel Analysis and Secure Design (COSADE) 2012*. Ed. by Werner Schindler and Sorin A. Huss. Vol. 7275. LNCS. Springer, 2012, pp. 33–38.

[Was03] Lawrence C. Washington. *Elliptic Curves. Number Theory and Cryptography*. Discrete Mathematics and its Applications. Chapman & Hall/CRC, 2003.

[Whe+09] Claire Whelan, Dan Page, Frederik Vercauteren, Michael Scott, and William Marnane. "Implementation Attacks & Countermeasures." In: *Identity-Based Cryptography*. Ed. by Marc Joye and Gregory Neven. Vol. 2. Cryptology and Information Security. IOS Press, 2009, pp. 226–243.

[Wri87] Peter Wright. *Spy Catcher. The Candid Autobiography of a Senior Intelligence Officer*. Heinemann, 1987.

[WS06] Claire Whelan and Michael Scott. "Side Channel Analysis of Practical Pairing Implementations: Which Path Is More Secure?" In: *Proceedings of Progress in Cryptology (VIETCRYPT) 2006*. Ed. by Phong Q. Nguyen. Vol. 4341. LNCS. Springer, 2006, pp. 99–114.

[WS07] Claire Whelan and Michael Scott. "The Importance of the Final Exponentiation in Pairings When Considering Fault Attacks." In: *Proceedings of Pairing-Based Cryptography — Pairing 2007*. Ed. by Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto. Vol. 4575. LNCS. Springer, 2007, pp. 225–246.

[Yen+03] Sung-Ming Yen, Seungjoo Kim, Seongan Lim, and Sang-Jae Moon. "RSA Speedup with Chinese Remainder Theorem Immune against Hardware Fault Cryptanalysis." In: *IEEE Transactions on Computers* 52.4 (2003), pp. 461–472.

[YJ00] Sung-Ming Yen and Marc Joye. "Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis." In: *IEEE Transactions on Computers* 49.9 (2000), pp. 967–970.