Improved Side Channel Attacks on Pairing Based Cryptography[*]

Johannes Blömer, Peter Günther, and Gennadij Liske

University of Paderborn, Germany
{johannes.bloemer,peter.guenther,gennadij.liske}@uni-paderborn.de

Abstract. Several known invasive and non-invasive attacks against pairing algorithms only work if the second but not if the first argument of the pairing is the secret. In this paper we extend some of these attacks to the case where the first argument is the secret. Hence we conclude that positioning the secret as the first argument of the pairing does not necessarily improve the security against side channel attacks (SCAs), as it sometimes has been suggested.

1 Introduction

Since the invention of the first fully functional identity based encryption (IBE) scheme [4], that was based on bilinear pairings, pairings have become an important tool in cryptography. Today numerous schemes such as hierarchical identity-based encryption, attribute based encryption, and identity based signatures make use of pairings as their building blocks [6]. 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 [17]. It is well known that in this case, mathematical cryptanalysis is not sufficient. Instead the vulnerability to side channel attacks (SCAs) has to be evaluated as well.

Bilinear pairings are usually realized on groups of elliptic curves. Although pairing based cryptography (PBC) uses methods from elliptic curve cryptography (ECC), the vulnerability of PBC against SCAs is not well understood. In ECC based schemes, such as ECDSA, the secret is a scalar multiplier of a point on the curve. In PBC, the secret is usually a point on the curve. Often this point is an argument of the pairing. Therefore, the pairing itself is an interesting target for SCAs. But when it comes to PBC the effort that has been spent on the analysis of SCAs is much smaller than in the case of standard ECC. Nevertheless, there are some results that analyze the vulnerability of pairings to passive attacks as well as to active attacks [12,8,21,11,22,13].

There is a variety of pairings that can be used for PBC, e.g., the Weil pairing, the Tate pairing, the eta pairing, and their variations. Obviously, SCAs depend heavily on the pairing and its specific implementation. Regarding non-invasive attacks,

[*] This work was funded by the German Ministry of Education and Research, grant 01IS10030C.

i.e., attacks that exploit timings, power consumption, or electro-magnetic radiance, in [8] the authors have investigated attacks for implementations of the eta pairing on supersingular curves in characteristic 2. In [21], differential power analysis (DPA) based attacks on the Tate pairing were analyzed. The authors showed how to attack implementations of the Tate pairing if the second argument represents the secret. The two arguments play different roles in the algorithms that compute the pairing. Hence, an attack on the second argument does not necessarily imply an attack on the first argument of the pairing. Furthermore, the authors of [21] conjectured that it may be more difficult to attack some implementations of the Tate pairing if the first argument of the pairing is the secret. In [11] this problem has been addressed for the case where the first argument is represented in Jacobian coordinates. Here, a DPA of a modular multiplication *and* a DPA of a modular addition was required to succeed.

In this paper we show that the attack from [21] that is based on a DPA of the modular multiplication can be extended to the case where the first argument is the secret. To achieve this, we assume that the first argument of the pairing is defined over the base field while the second argument is defined over the extension field. This setting is relevant for many efficient implementations [2]. Furthermore, for the attack from [11] that needs a DPA of a modular multiplication and a DPA of an addition we show that the first one is already sufficient. To do this we need to limit the attack to an early iteration of the Miller algorithm. We further show that twists, often used for more efficient implementations, entail further possibilities to attack bilinear pairings with a DPA.

With respect to invasive SCAs, the first result in the context of PBC was presented in [13]. The authors attacked two algorithms for the Tate pairing. Later, the vulnerability of several algorithms for the Weil, Tate and eta pairings in presence of fault attacks was studied in [22]. The authors of [10] analyze the vulnerability of Miller's algorithm against the same fault type as used in [13]. To apply these results to concrete pairing algorithms that are based on Miller's algorithm, stronger assumptions about the induced faults are necessary than in [13] and [22].

In this paper we are especially interested in the fault attacks from [22] on an algorithm for the eta pairing. One of the most realistic attacks in [22] is only possible if the secret is used as the second argument of the pairing. We extend this attack in two directions. First we show how a more general type of fault can be handled. Furthermore we show that our approach can handle a secret first argument just as well as a secret second argument.

Altogether, we conclude that schemes where the first argument is the secret are not less vulnerable to non-invasive and invasive SCAs than schemes where the second argument is the secret.

The work is organized as follows. In Section 2, we introduce the necessary background on elliptic curve cryptography and pairing based cryptography. In Section 3, we present one of our main results, namely a non-invasive attacks on an implementation of the Tate pairing when the first argument is secret. We do this by means of a DPA of the modular multiplication. A DPA for the multiplication has been described in [21] that we use in a different manner. We will also consider two approaches to circumvent our attacks. In Section 4 we generalize the fault attacks of [22] and show that they are as powerful in the case where the first argument of the pairing is the secret as in the case where the second argument is the secret.

2 Background

Before we will provide some background about ECC and especially PBC we will introduce the notation that we follow throughout this work.

2.1 Notation

By $\mathbb{F}_{q^e}$ we denote a field of size q^e. For q prime, we always assume that elements of $\mathbb{F}_q$ are stored in the binary representation of their representatives in $\mathbb{Z}_q$. We view the degree e extension $\mathbb{F}_{q^e} = \mathbb{F}_q(\alpha)$ of $\mathbb{F}_q$ as an e-dimensional vector field with basis $\{1, \alpha, \ldots, \alpha^{e-1}\}$ and emphasize this by writing elements $a \in \mathbb{F}_{q^e}$ as $a = \sum_{i=0}^{e-1} a^{(i)} \alpha^i$ with $a^{(i)} \in \mathbb{F}_q$.

For a polynomial $f \in \mathbb{F}_q[x, y]$ we denote the degree of f in x and y by $\deg_x f$ and $\deg_y f$, respectively.

2.2 Elliptic Curve Arithmetic

For fields $\mathbb{F}_q$ of characteristic $p > 3$ we consider elliptic curves defined over $\mathbb{F}_q$ in short Weierstrass form:

$$E : F(x, y) = y^2 - x^3 - a_4 x - a_6 \text{ with } a_4, a_6 \in \mathbb{F}_q. \tag{1}$$

We write $E(\mathbb{F}_{q^e})$ for the $\mathbb{F}_{q^e}$ rational points on E, i.e. for points with coordinates in the field $\mathbb{F}_{q^e}$.

It is well known that an additive group can be defined on $E(\mathbb{F}_{q^e})$ with the point at infinity $\mathcal{O}$ as the neutral element [19]. The addition of two points $R, P \in E$ is based on the computation of the line trough $R = (x_R, y_R)$ and $P = (x_P, y_P)$ with slope $\lambda_{R,P}$. The line intersects E at a third point $-(R + P)$. For doubling R the tangent to E at R is computed instead. It has slope $\lambda_{R,R}$ and intersects the curve at $-2R$:

$$\lambda_{R,P} = \frac{y_P - y_R}{x_P - x_R} \qquad \lambda_{R,R} = \frac{3x_R^2 + a_4}{2y_R}. \tag{2}$$

With the double and add algorithm, the scalar multiplication aP can be calculated efficiently. The subgroup of the a-torsion points $E(\mathbb{F}_q)[a] \subseteq E(\mathbb{F}_q)$ is defined as the kernel of the multiplication by a.

In the following sections we will frequently be given a polynomial $f(x, y) \in \mathbb{F}_q[x, y]$ of fairly small degree. We will be interested in the zeros of f that are points on a given elliptic curve. The next theorem based on the proof of [19, Corollary 2.3.1] shows that this problem can be solved.

Theorem 1. *Given a finite field $\mathbb{F}_q$ of characteristic $p > 3$, let $f(x,y) \in \mathbb{F}_q[x,y]$ be a polynomial of total degree n. Furthermore, let E be an elliptic curve defined over $\mathbb{F}_q$ as in* (1)*. Then there is an algorithm that finds all zeros of $f(x,y)$ on $E(\mathbb{F}_q)$ in expected running time $O\left(n^3 \log q\right)$.*

Proof. Consider the polynomial $\hat{f}(x,y) = f(x,y)\, f(x,-y)$ that we write as $\hat{f}(x,y) = f_1\left(x,y^2\right) + y f_2\left(x,y^2\right)$ for $f_1, f_2 \in \mathbb{F}_q\left[x,y^2\right]$. Since $\hat{f}$ is defined such that $\hat{f}(x,y) = \hat{f}(x,-y)$ we obtain $y f_2\left(x,y^2\right) = -y f_2\left(x,y^2\right)$. With the characteristic of $\mathbb{F}_q$ not equal to 2 it holds that $f_2\left(x,y^2\right) = 0$. Hence, $\hat{f} \in \mathbb{F}_q\left[x,y^2\right]$. With $F(x,y)$ from (1) we can replace y^2 by $x^3 + a_4 x + a_6$. Hence, we get a univariate polynomial $\tilde{f} \in \mathbb{F}_q[x]$ that is equivalent to $\hat{f}$ in $\mathbb{F}_q[x,y] / (F(x,y))$.

From the definition of $\hat{f}$ it follows that $\deg_x \hat{f} \leq 2 \deg_x f$ and $\deg_y \hat{f} \leq 2 \deg_y f$. Since we replace y^2 by $x^3 + a_4 x + a_6$ during the transformation of $\hat{f}\left(x,y^2\right)$ to $\tilde{f}(x)$ it holds that $\deg_x f \leq 2 \deg_x f + 3 \deg_y f \leq 3n$.

Finally, for $(\alpha,\beta) \in E(\mathbb{F}_q)$ and $f(\alpha,\beta) = 0$ it holds that $\hat{f}(\alpha,\beta) = 0$ and hence $\tilde{f}(\alpha) = 0$. Finding α as an element of the $\mathbb{F}_q$-rational roots of $\tilde{f}$ can be done with probabilistic algorithms like the Cantor-Zassenhaus factoring algorithm in expected running time $O\left(\left(\deg_x \tilde{f}\right)^3 \log q\right)$ [18]. From $F(x,y)$ and α, the corresponding points (α,β) and $(\alpha,-\beta)$ can be computed and f can be tested for $f(\alpha,\pm\beta) = 0$ to return β.

□

2.3 Pairing Based Cryptography

The key element of pairing based cryptography is a bilinear map called pairing. There are different pairings known in the literature, e.g. the Weil pairing and the Tate pairing. Later in this section, we will introduce the Tate pairing and an implementation that is based on the Miller algorithm [9]. We will then introduce another pairing for the special case of characteristic 2, called the eta pairing [14].

For pairing based schemes, their implementation usually offers some degree of freedom with respect to the arguments of the pairing. For example the secret key can be chosen as either the first or the second argument. We are interested in the case where the secret key is chosen as the first argument. In [21] it was conjectured that this makes passive SCAs more difficult. We will show in Section 3 that this is not necessarily the case.

Definition of the Tate Pairing. We will now define the reduced Tate pairing with final exponentiation. Let $P \in E(\mathbb{F}_q)[l]$ and let $f_{l,P}$ denote a function on E with divisor $(f_{l,P}) = l\,(P) - l\,(\mathcal{O})$ [19]. Let k be the embedding degree (see [19]) of l with respect to q. With $k > 1$, groups $\mathbb{G}_1 = E(\mathbb{F}_q)[l]$ and $\mathbb{G}_2 \subseteq E(\mathbb{F}_{q^k})/lE(\mathbb{F}_{q^k})$, the *simplified* Tate pairing [2] is defined as:

$$\begin{aligned} \mathrm{e} : \mathbb{G}_1 \times \mathbb{G}_2 &\to \mathbb{F}_{q^k}^{*} \\ (P,Q) &\mapsto f_{l,P}(Q)^{(q^k-1)/l}. \end{aligned} \tag{3}$$

Note that $\mathbb{G}_1$ is restricted to the l-torsion points that are defined over $\mathbb{F}_q$ and that $k > 1$. This specific choice was made in [2] because it allows the efficient computation of the pairing in (3) which invokes only one evaluation of $f_{l,P}$. To ensure non-degeneracy in this case, the second argument $\mathbb{G}_2$ has to be selected such that $\mathbb{G}_2 \nsubseteq E(\mathbb{F}_q)[l]$ [3, Lemma IX.8]. We will later make use of this common setup to apply a passive SCA.

Implementation of the Tate Pairing for General Characteristics. In this section we recall how the Tate pairing from (3) can be computed efficiently for finite fields $\mathbb{F}_q$. Even though the degree of $f_{l,P}$ is exponential in $\log(l)$, V. Miller

Algorithm 1. Miller Algorithm for evaluating a function $f_{l,P}$ with divisor $(f_P) = l\,(P) - l\,(\mathcal{O})$ at Q. The function $g_{U,V}(Q)$ is defined in (4).

```
Require: $P \in E[l](\mathbb{F}_q), Q = (x_Q, y_Q) \in E(\mathbb{F}_{q^k})$, binary representation $l = (l_{t-1} \dots l_0)$
Ensure: $f_{l,P}(Q)$ where $(f_{l,P}) = l\,(P) - l\,(\mathcal{O})$
 1: procedure $f_{l,P}(Q)$
 2:     $f \leftarrow 1, v \leftarrow 1, R \leftarrow P$
 3:     for $j \leftarrow t-1, \dots, 1$ do
 4:         $f \leftarrow f^2 \cdot g_{R,R}(x_Q, y_Q)$, $v \leftarrow v^2 \cdot g_{2R,-2R}(x_Q, y_Q)$
 5:         $R \leftarrow 2R$
 6:         if $l_j = 1$ then
 7:             $f \leftarrow f \cdot g_{R,P}(x_Q, y_Q)$, $v \leftarrow v \cdot g_{R+P,-(R+P)}(x_Q, y_Q)$
 8:             $R \leftarrow R + P$
 9:         end if
10:     end for
11:     return $f/v$
12: end procedure
```

introduced an efficient algorithm that iteratively computes the value of $f_{l,P}$ at Q [9]. The running time of this algorithm is only linear in $\log(l)$. For the details see Algorithm 1.

In order to construct the value of $f_{l,P}(Q)$, lines through R, V, and $-(R+V)$ with $V \in \{P, R\}$ are computed:

$$g_{R,V}(x,y) = y - y_{-(R+V)} - \lambda_{R,V}(x - x_{-(R+V)}). \tag{4}$$

The slope $\lambda_{R,V}$ is defined as in (2). Also note that with restricting $\mathbb{G}_1$ to points in $E(\mathbb{F}_q)[l]$ the computation of the Miller algorithm becomes more efficient. For example the point doubling and the point addition in Line 5 and in Line 8 of the algorithm only requires arithmetic in $\mathbb{F}_q$ instead of arithmetic in $\mathbb{F}_{q^k}$.

Remark 1. In any iteration j, the point R of Algorithm 1 equals aP for some $a \in \mathbb{Z}_l$. If j is known, the value a can be efficiently computed from j. Since l is a prime, we can recover P from R with

$$P = (a^{-1} \mod l)R. \tag{5}$$

Now, we extend the setting to Jacobian coordinates in order to avoid the costly division in (2). Then, the affine representation $(x, y) \in E(\mathbb{F}_q)\backslash\{\mathcal{O}\}$ is mapped to the equivalence class $(X : Y : Z) = (xZ^2 : yZ^3 : Z)$ with $Z \in \mathbb{F}_q^*$. With the Jacobian representation $R + P$ as well as $2R$ in Algorithm 1 can be computed without the costly division of (2):

$$\lambda_{R,P} = Y_P Z_R^3 - Y_R Z_P^3 \qquad\qquad \lambda_{R,R} = 3X_R^2 + a_4 Z_R^4. \tag{6}$$

As a consequence, the line function (4) has to be adapted:

$$g_{R,V}(x, y) = yZ^3_{-(R+V)} - Y_{-(R+V)} - \lambda_{R,V}\left(xZ^2_{-(R+V)} - X_{-(R+V)}\right) \tag{7}$$

with $V \in \{R, P\}$.

The Eta Pairing for Characteristic 2. The eta pairing was defined in [14] for different types of algebraic curves. In [20] the authors introduced an efficient method and an algorithm to compute this pairing for supersingular elliptic curves. C. Whelan and M. Scott [22] presented a slightly modified version of this algorithm for characteristic 2 (Algorithm 2). Both input points are in $E(\mathbb{F}_{2^m})[l]$, whereas the output of the pairing is an element of the multiplicative group of $\mathbb{F}_{2^{4m}} = \mathbb{F}_{2^m}(\alpha) = \mathbb{F}_{2^m}[x]/(x^4 + x + 1)$, where α is a zero of $x^4 + x + 1$.

Algorithm 2. Algorithm to compute the eta pairing without final exponentiation

```
Require: $P = (x_P, y_P), Q = (x_Q, y_Q) \in E(\mathbb{F}_{2^m})[l]$
Ensure: $\eta(P, Q) \in \mathbb{F}_{2^m}^*(\alpha)$
 1: $g \leftarrow 1, v \leftarrow 1, T \leftarrow P$
 2: for $j \leftarrow m - 1$ to $0$ do
 3:     $\lambda_j \leftarrow x_T^2 + 1$
 4:     $g_j \leftarrow (y_Q + y_T + \lambda_j(x_Q + x_T + 1)) + (\lambda_j + x_Q + 1)\alpha + (\lambda_j + x_Q)\alpha^2$
 5:     $g \leftarrow g^2 \cdot g_j$
 6:     $T \leftarrow 2T$
 7:     $v_j \leftarrow (x_Q + x_T + 1) + \alpha + \alpha^2$
 8:     $v \leftarrow v^2 \cdot v_j$
 9: end for
10: return $g/v$
```

3 Passive Attack of the Tate Pairing with Secret P

In this section we consider the Tate pairing $\mathrm{e}(P, Q)$ from Section 2.3. We extend the attack from [21] to the case where the first argument P is the secret. As in [21] we also use a DPA of the modular multiplication as the basic tool. First, we will introduce an attack based on the computation of (4) for affine coordinates. Then, we will show an attack based on Jacobian coordinates that complements

Table 1. Summary of our passive attacks based on a DPA of the modular multiplication. Here, d denotes the twist degree (see Section 3.4 for some background on twists) and k denotes the embedding degree of the pairing implementation.

Section	Target Multiplication	Restrictions
3.2	$\lambda_{R,R} \cdot (x_Q - x_{-2R})$	does not apply for the case $d = k = 2$
3.3	$Z_{-2R}^2 \cdot x_Q$ or $Z_{-2R}^3 \cdot y_Q$	an early iteration of Alg. 1
3.4	$y_Q \cdot (-y_{-2P} + \lambda_{P,P} x_{-2P})$	twists of degree 4 or 6, 1st iteration of Alg. 1

the results of [11] by getting rid of the additional DPA of the modular addition. Finally, we will show an additional possibility to attack a pairing when its implementation is based on twists of degree $d \in \{4, 6\}$. Twists are isomorphic curves that enable faster implementations [5]. Table 1 summarizes the main results of this section.

We are interested in the implementation of a pairing from Algorithm 1 and we assume $P \in E(\mathbb{F}_q)$ and $Q \in E(\mathbb{F}_{q^k})$, $Q \notin E(\mathbb{F}_q) \setminus \{\mathcal{O}\}$. This assumption was justified in Section 2.3. The most important reason is that it offers very efficient implementations of a non-degenerate pairing. For clarity we restrict ourselves to fields of large prime characteristic. Nevertheless, at least the results of Section 3.2 are directly applicable also to fields of characteristic 2 or 3.

3.1 DPA of Modular Multiplication

In [21], the modular multiplication $a \cdot b$ of elements in $\mathbb{F}_q$ was exploited to leak information about one argument of the multiplication. Assume our goal is to learn a. As one part of the DPA we have to predict the result of $a \cdot b$ for different hypotheses of a. Hence it is a requirement for a successful DPA to be able to freely choose the other argument b of the multiplication. In the following sections we will show how it is possible to satisfy this requirement during the computation of the Tate pairing and consequently how to recover the secret argument of the pairing.

3.2 Generic Attack Based on Affine Coordinates

Although Jacobian coordinates are often assumed to be more efficient than affine coordinates, recent results [1] indicate that the latter become more relevant for modern processors. Among other things, this motivates our analysis of affine coordinates with respect to SCAs. To apply our attack, we will use the fact that P is already defined over $\mathbb{F}_q$, as explained in Section 2.3. Our attack is in two steps. First we give definitions for a and b that fulfill the requirements of the DPA. In our case the DPA will not give us the secret point P directly. Instead it will result in an element a of $\mathbb{F}_q$ that is related to P. Thus, in a second step we show how P can be recovered from a.

Defining the Targets $\boldsymbol{a}$ and $\boldsymbol{b}$ of the DPA. As in [21] we will use the computation of the function $g_{R,V}(x,y)$ from (4) as the target of our attack. Recall the outline of Miller's algorithm from Section 2.3. With the notation introduced there, we fix an iteration j and write the value of the point R in Line 4 as $R = (x_R, y_R)$. The function $g_{R,V}(x,y)$ is evaluated with $V = R$, $x = x_Q$, and $y = y_Q$. Note that R is a multiple of the secret P. We now insert R, x_Q, and y_Q into (4) and get

$$g_{R,R}(x_Q, y_Q) = y_Q - y_{-2R} - \lambda_{R,R}(x_Q - x_{-2R}) \tag{8}$$

with $\lambda_{R,R}$ from (2) as calculated during the doubling of R in iteration j. Notice that $\lambda_{R,R}$ is multiplied with $(x_Q - x_{-2R})$. Since we assumed $P \in E(\mathbb{F}_q)$ it follows that $x_{-2R} \in \mathbb{F}_q$ and with our definition of extension fields we can write x_{-2R} as $x_{-2R} = x_{-2R}\alpha^0 + 0\alpha^1 + \cdots + 0\alpha^{k-1} \in \mathbb{F}_{q^k}$. We further assume that either no twist (see Section 3.4) of degree $d = 2$ is used or that the embedding degree k is larger than 2. Then $Q \notin E(\mathbb{F}_q) \backslash \{\mathcal{O}\}$ implies $x_Q \notin \mathbb{F}_q$ [5]. Hence, x_Q is of the form $x_Q = \sum_{i=0}^{k-1} x_Q^{(i)} \alpha^i$ and there exists an $i \geq 2$ such that $x_Q^{(i)} \neq 0$. Therefore, we get $x_Q - x_{-2R} = \left(x_Q^{(0)} - x_{-2R}\right)\alpha^0 + \sum_{i=1}^{k-1} x_Q^{(i)}\alpha^i$. In settings of DPA we set $a = \lambda_{R,R}$ and $b = x_Q^{(i)}$ for $x_Q^{(i)} \neq 0$. Hence, a only depends on P, while b only depends on Q. The latter is under our control.

Recovery of $\boldsymbol{P}$ from $\boldsymbol{\lambda_{R,R} = a}$. According to (2), we can consider $\lambda_{R,R}$ as a rational function in the coordinates (x_R, y_R) of R. Hence, we can clear the denominator of $\lambda_{R,R} - a$ to obtain the polynomial $f(x_R, y_R) = (3x_R^2 + a_4) - 2y_R a$ with $\deg_{x_R} f = 2$ and $\deg_{y_R} f = 1$. If $\lambda_{R,R}$ was correctly recovered (i.e. $\lambda_{R,R}(x_R, y_R) = a$), then this polynomial has at least one zero in $E(\mathbb{F}_q)$, namely R. According to Theorem 1 all zeros can be determined in expected polynomial time of $O(\log q)$. Furthermore, there are at most four possible candidates. If the iteration j is known, these solutions directly translate to at most four candidates for P by applying (5). Now, P can be found by testing all candidates. For example in an encryption scheme, the ciphertext of an arbitrary message could be decrypted with all candidates. Then the point P that leads to the original message is the correct key. If otherwise, the iteration j is not known, we can still try all $\log(l)$ possibilities for j.

Summary of the Attack. To summarize, we made use of a reasonable restriction of the arguments of the pairing in the sense of efficient implementations: $P \in E(\mathbb{F}_q)$, $Q \notin E(\mathbb{F}_q)$. This enabled us to find a coefficient of $x_Q - x_{-2R} \in \mathbb{F}_{q^k}$ in (4) that does not depend on P. Based on this we were able to apply a DPA similar to the one presented in [21] also to the case where the first argument P is secret.

3.3 Attack Based on Projective Coordinates

As in the previous attack we will apply the DPA to the computation of (4) in Algorithm 1. But this time, we will look at the representation of the variable point R in Jacobian coordinates. We assume that the input P is normalized such that $P = (x_P : y_P : 1)$. These mixed coordinates are common for efficient implementations because they save multiplications with Z_P in (7) at every iteration that performs the addition with P [15]. While the attack of [11] on Jacobian coordinates requires to attack one modular multiplication *and* one addition, the assumption $Z_P = 1$ allows us to avoid the attack on the addition. However, our approach is restricted to one of the early iterations of Algorithm 1.

Defining the Targets a and b of the DPA. Again, we fix an iteration j of Algorithm 1 and analyze the doubling step in Line 4 of Algorithm 1. But this time for the version in Jacobian coordinates from (7) with $V = R$. Since P is the secret argument of the pairing, as always we assume that $Q = (x_Q, y_Q)$ is under our control. Then there are two obvious operations to attack with the tools from Section 3.1: $x_Q \cdot Z^2_{-2R}$ and $y_Q \cdot Z^3_{-2R}$. We describe the case where we decide to attack $x_Q \cdot Z^2_{-2R}$, but $y_Q \cdot Z^3_{-2R}$ can be approached in the same way. With $P \in E(\mathbb{F}_q)$, $Q \in E(\mathbb{F}_{q^k})$, and $\mathbb{F}_{q^k} = \mathbb{F}_q(\alpha)$, we get $Z_{-2R} \in \mathbb{F}_q$, $x_Q = \sum_{i=0}^{k-1} x_Q^{(i)} \alpha^i$, $x_Q^{(i)} \in \mathbb{F}_q$. We observe that $x_Q^{(0)}, \dots, x_Q^{(k-1)}$ can be controlled so we choose $a = Z^2_{-2R}$ and $b = x_Q^{(i)}$ for a fixed i as input of the DPA. If the DPA is successful, we obtain the value $Z^2_{-2R} = a$ for iteration j.

If we decided to attack $y_Q \cdot Z^3_{-2R}$ we get $Z^3_{-2R} = a$ as the result of the DPA. Hence we proceed with Z^τ_{-2R} and $\tau \in \{2, 3\}$.

Recovery of P from $Z^\tau_{-2R} = a$. The point R in iteration j is obtained by repeated application of the addition and doubling formulas for Jacobian coordinates to P. These formulas define a polynomial mapping for points on the curve (1). For efficiency, we assumed the representation $P = (x_P : y_p : 1)$ as explained above. Therefore, Z_{-2R} can be expressed as a polynomial in x_P and y_P. Hence we are interested in points on the curve (1) that are roots of the polynomial $Z^\tau_{-2R}(x_P, y_P) - a$. The problem lies in the degree of $Z^\tau_{-2R}(x_P, y_P)$ that is exponential in the attacked iteration j. But if j is not too large this polynomial can be constructed recursively from the formulas for doubling and adding points in Jacobian coordinates. Furthermore, we can apply Theorem 1 from Section 2.2 to find all candidates for P. In our simulation with a Pari/GP based implementation, we were able to recover P from Z^τ_{-2R} for $j \leq 5$ and fields of size $\log(q) = 1024$ in a few minutes. As described in Section 3.2 we can try all resulting candidates for P and also j to find the correct one.

Summary of the Attack. In summary, we described how to apply the DPA on the multiplication to recover the Z-coordinate of an intermediate point. We further 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 1 is attacked. We thus can avoid to additionally attack the modular addition like it was required in [11].

3.4 Attack Based on Twists

In this section we explain how the structure imposed by so-called twists gives us further possibilities to attack pairings with secret in the first argument. Following the same strategy as in the previous sections, we will point to other invocations of a modular multiplication that can be attacked and therefore need to be considered when securing the implementation of pairings against SCAs. First we give the required background on twists.

Background on Twists. As explained in Section 2.3, we are considering implementations of the Tate pairing $\mathrm{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{F}_{q^k}^*$ from (3) on the $\mathbb{F}_{q^k}$-rational points of an elliptic curve E with $\mathbb{G}_2 \subseteq E(\mathbb{F}_{q^k})$. For some curves E there exists a curve E', an integer $d|k$, and a map $\psi : E' \to E$ with the following properties [5]:

1. ψ defines an isomorphism from $E'(\mathbb{F}_{q^k})$ to $E(\mathbb{F}_{q^k})$
2. $E(\mathbb{F}_{q^k})[l] \simeq E(\mathbb{F}_q)[l] \oplus \psi(E'(\mathbb{F}_{q^{k/d}})[l])$
3. ψ can be computed efficiently.

Hence, we can choose $\mathbb{G}_2 = \psi(E'(\mathbb{F}_{q^{k/d}})[l])$ and we call E' a *degree d twist* of E. This enables more efficient implementations because arithmetic in $\mathbb{G}_2$ can be moved to the subgroup $E'(\mathbb{F}_{q^{k/d}})[l]$ that is already defined over the smaller field $\mathbb{F}_{q^{k/d}}$. Naturally, the use of twists is a common optimization for the computation of the Tate pairing. Furthermore, for fields of characteristic not equal to 2 or 3 it holds that $d \in \{2, 3, 4, 6\}$ [19]. As explained in [5], one often chooses $\mathbb{F}_{q^k}$, E, E', and ψ such that $\mathbb{F}_{q^k} = \mathbb{F}_{q^{k/d}}(\alpha)$ and

$$\begin{aligned} \psi : E'(\mathbb{F}_{q^{k/d}}) &\to E(\mathbb{F}_{q^k}) \\ (x, y) &\mapsto (\alpha^2 x, \alpha^3 y). \end{aligned} \tag{9}$$

Assuming this definition, the map ψ can be implemented more efficiently. This is because a multiplication of $x, y \in \mathbb{F}_{q^{k/d}}$ with $\alpha^i \in \mathbb{F}_{q^k}$, $i \in \{2, 3\}$ simply selects the index $i \mod d$ of x and y of the d-dimensional vectors representing $\psi(x, y) \in \mathbb{F}_{q^k}$. For $i \geq d$, only an additional multiplication with α^d in $\mathbb{F}_{q^{k/d}}$ is required.

Defining the Targets *a* and *b* of the DPA. From (9) we see that ψ imposes an additional structure on the representation of group elements in $\mathbb{G}_2$. For $d \in \{4, 6\}$ we show how this structure enables further possibilities to recover the secret point P by means of the DPA from Section 3.1. We assume that $\mathbb{F}_{q^k}$ is represented by the basis $\{1, \alpha, \ldots, \alpha^{d-1}\}$. Hence, α^2 and α^3 are not in $\mathbb{F}_{q^{k/d}}$. We target the very first iteration of Algorithm 1 where we obtain the following value after the execution of Line 4:

$$g_{P,P}(\psi(x_Q, y_Q)) = (-y_{-2P} + \lambda_{P,P} x_{-2P}) + 0\alpha - \lambda_{P,P} x_Q \alpha^2 + y_Q \alpha^3 \in \mathbb{F}_{q^k}. \tag{10}$$

Note that depending on the second most significant bit (MSB) of l, this value will be multiplied with $g_{P,2P}(x_Q, y_Q)$ in Line 7 or squared in Line 4 of the second iteration. The outline of the attack is the same for both cases, so we look at the latter case where we square (10). This squaring in $\mathbb{F}_{q^k}$ involves a multiplication in $\mathbb{F}_{q^{k/d}}$ of y_Q with $-y_{-2P} + \lambda_{P,P} x_{-2P}$. Consequently, we can again apply the DPA of the modular multiplication with $a = -y_{-2P} + \lambda_{P,P} x_{-2P}$ and $b = y_Q$ to recover $a = -y_{-2P} + \lambda_{P,P} x_{-2P}$. We are able to express x_{-2P}, y_{-2P}, and $\lambda_{P,P}$ as functions of x_P and y_P. After clearing the denominator of $\lambda_{P,P}$ in (2) we obtain the polynomial $f(x_P, y_P) = -2y_P y_{-2P} + (3x_P^2 + a_4)x_P - a2y_P$. From this polynomial we can efficiently recover the root P by Theorem 1.

Summary of the Attack. To summarize, we assumed an implementation based on twists of degree 4 or 6. In this case, the map ψ from the twisted curve to the original curve allowed us to separate $-y_{-2P} + \lambda_{P,P} x_{-2P}$ from y_Q in (10). This gave us the possibility to apply a DPA at an additional location compared to the previous sections.

3.5 Countermeasures

Several countermeasures like point blinding [13], randomized projective coordinates [8], and randomization of intermediate values [16] have been proposed to secure the Tate pairing against SCAs. We will now recall two of them that also protect against the attacks of the previous section.

We slightly modify the approach of the additive blinding from [13] and initially share the secret P as $P = P_1 + P_2$ in a secure environment with P_1 uniformly from $\mathbb{G}_1$. Then we make use of the bilinearity and compute the pairing as

$$\mathrm{e}(P, Q) = \mathrm{e}(P_1, Q)\, \mathrm{e}(P_2, Q).$$

Before the next invocation of the pairing computation we refresh the shares by computing $P_1' = P_1 + T$ and $P_2' = P_2 - T$ for uniform $T \in \mathbb{G}_1$. The attacks we described in the previous sections are all defeated by this countermeasure.

To save one exponentiation with $(q^k - 1)/l$ we actually propose the following implementation

$$\mathrm{e}(P, Q) = \left(f_{l,P_1}(Q) f_{l,P_2}(Q)\right)^{(q^k-1)/l}.$$

On the one hand, this technique is relatively expensive because it requires two runs of Algorithm 1 instead of one. On the other hand, it is quite general. The reason is that the computation of the pairing is secured against all first order DPAs, i.e. DPAs that are restricted to one operation of the overall computation. Another advantage is that the computation of $f_{l,P}$ by Algorithm 1 itself does not need to be adapted. This makes it easy to seamlessly switch between fast but unprotected implementations and protected but slower implementations. Further note that the approach of sharing P can be generalized to n shares in order to defeat against attacks of order $n - 1$.

Another approach to circumvent our attacks are randomized projective coordinates [8]. Here, the implementation of the pairing has to be based on Jacobian coordinates as described in Section 3.3. But now we drop the assumption that the argument P of the pairing is normalized to $P = (x_P : y_P : 1)$. Instead we choose Z uniformly at random from $\mathbb{F}_q$ before the computation of the pairing and set $P = (x_P Z^2 : y_P Z^3 : Z)$. Hence Z^{τ}_{-2R}, the target of the DPA, is randomized. This prevents the specific attack on Jacobian coordinates because the target is changing between two invocations of the pairing. The additional costs for this countermeasure are one squaring and four multiplications in the addition of $R + P$ in Line 7 of Algorithm 1 [15, Chapter 13.2].

4 Fault Attack on Pairing Based Cryptography

Until now we only considered passive attacks. In this section we will look at fault attacks. Here, the adversary corrupts the computation of the cryptographic algorithm to produce invalid outputs that allow the recovery of the secret key. In the context of PBC a few theoretical results are known [13,22,10]. See also [6, Chapter 14] or [7, Chapter 13] for an overview.

In [22] the authors considered fault attacks against several pairing algorithms under the assumption that random faults can be produced at particular memory cells. They concluded that pairings without final exponentiation are more vulnerable against this type of attacks. In particular, fault attacks for the eta pairing without final exponentiation were presented (see Algorithm 2 for details). The algorithm computes the eta pairing iteratively as

$$\eta\,(P, Q) = \frac{\prod_{i=1}^{m} \left(g_{m-i}(Q)\right)^{2^{m-i}}}{\prod_{i=1}^{m} \left(v_{m-i}(Q)\right)^{2^{m-i}}}. \tag{11}$$

The main idea behind the fault attacks of [22] is to attack one single iteration in order to isolate a factor of (11) in the first step. In the second step the secret is recovered from this factor. Recall the special form of $g_j(Q)$ in Algorithm 2:

$$g_j(Q) = g_j(Q)^{(0)} + g_j(Q)^{(1)}\alpha + \left(1 + g_j(Q)^{(1)}\right)\alpha^2 \tag{12}$$

where α is the defining element of the extension $\mathbb{F}_{2^{4m}}$ of $\mathbb{F}_{2^m}$ (see Section 2.1) and

$$g_j(Q)^{(0)} = y_Q + y_T + \lambda_j(x_Q + x_T + 1) \qquad g_j(Q)^{(1)} = \lambda_j + x_Q + 1. \tag{13}$$

The two most realistic attacks from [22] consider a fault at $g_0(Q)^{(0)}$ or $v_0(Q)^{(0)}$, respectively. But for the first case, the authors were able to recover the secret only if it is Q. We consider attacks on the computation of $g_j(Q)$ and extend [22] in two directions. At first we show how to handle additional locations for the faults in a unified way. Secondly, our approach equally allows recovery of secret P or secret Q.

We consider arbitrary faults at x_Q, y_Q, x_T, y_T, λ_j, $g_j(Q)^{(0)}$, and $g_j(Q)^{(1)}$ that affect only one iteration. For example a temporary fault in iteration j on x_Q may be injected while loading this fixed value into the CPU registers. As we can see in (12), all these examples have in common that the modification of $g_j(Q)^{(1)}$ equally applies to $g_j(Q)^{(2)} = g_j(Q)^{(1)} + 1$. This motivates our analysis of the case where $g_j(Q)$ is modified into $g_j(Q) + \Delta$ with $\Delta = \Delta^{(0)} + \Delta^{(1)}\left(\alpha + \alpha^2\right)$. In [22] only faults of $g(Q)^{(0)}$ were considered. Note that our model implicitly covers this case with $\Delta^{(1)} = 0$. A more careful analysis of the special structure of $g_j(Q)$ from Algorithm 2 enables our generalization as we will explain now.

Recovery of a Single Factor g_j. We start with the correct result of the pairing $\eta(P,Q)$. Then we execute the pairing computation to mount a fault attack. As explained above, we assume a fault that modifies $g_j(Q)$ into $g'_j(Q) = g_j(Q) + \Delta$ and results in the faulty result $\eta(P,Q)'$. Dividing the result of the corrupted execution by the result of the uncorrupted execution we obtain:

$$U_j := \frac{\eta(P,Q)'}{\eta(P,Q)} = \left(\frac{g_j(Q) + \Delta}{g_j(Q)}\right)^{2^j}. \tag{14}$$

All other factors from (11) are canceled.

With $U_j = ((A + \Delta)/A)^{2^j}$, where $A = a^{(0)} + a^{(1)}\alpha + \left(1 + a^{(1)}\right)\alpha^2$ we are able to describe all solutions of (14) that explain the observed value of U_j. First note that square roots are unique in characteristic 2. This allows us to write (14) as $A\left(1 + \sum_{i=0}^{3} u^{(i)}\alpha^i\right) = \Delta = \Delta^{(0)} + \Delta^{(1)}\left(\alpha + \alpha^2\right)$ with $U_j^{2^{-j}} = \sum_{i=0}^{3} u^{(i)}\alpha^i$. Expanding the left side and applying the equivalence $\alpha^4 = \alpha + 1$ in $\mathbb{F}_{2^{4m}}$ gives us a system of four equations over $\mathbb{F}_{2^m}$ with one equation for each coefficient and variables $\Delta^{(0)}$, $\Delta^{(1)}$, $a^{(0)}$, and $a^{(1)}$:

$$\begin{pmatrix} 1 & 0 & u^{(0)} + 1 & u^{(2)} + u^{(3)} \\ 0 & 1 & u^{(1)} & 1 + u^{(0)} + u^{(2)} \\ 0 & 1 & u^{(2)} & 1 + u^{(0)} + u^{(1)} + u^{(3)} \\ 0 & 0 & u^{(3)} & u^{(1)} + u^{(2)} \end{pmatrix} \begin{pmatrix} \Delta^{(0)} \\ \Delta^{(1)} \\ a^{(0)} \\ a^{(1)} \end{pmatrix} = \begin{pmatrix} u^{(2)} \\ u^{(2)} + u^{(3)} \\ 1 + u^{(0)} + u^{(3)} \\ u^{(1)} \end{pmatrix}.$$

Adding the second line to the third line leaves us with the following system for $a^{(0)}$ and $a^{(1)}$:

$$\begin{pmatrix} u^{(1)} + u^{(2)} & u^{(1)} + u^{(2)} + u^{(3)} \\ u^{(3)} & u^{(1)} + u^{(2)} \end{pmatrix} \begin{pmatrix} a^{(0)} \\ a^{(1)} \end{pmatrix} = \begin{pmatrix} 1 + u^{(0)} + u^{(2)} \\ u^{(1)} \end{pmatrix}. \tag{15}$$

This system has at least one solution, namely $a^{(0)} = g_j(Q)^{(0)}$ and $a^{(1)} = g_j(Q)^{(1)}$. Actually, this is the only solution because of the special structure of $a^{(0)} = g_j(Q)^{(0)}$ and Δ. To see this, consider another solution B, Γ of (14) with the same structure as A and Δ. From $U_j^{2^{-j}} = (A + \Delta)/A = (B + \Gamma)/B$ we get

$$\begin{aligned} &\left(b^{(0)} + b^{(1)}\alpha + \left(1 + b^{(1)}\right)\alpha^2\right)\left(\Delta^{(0)} + \Delta^{(1)}\left(\alpha + \alpha^2\right)\right) \\ &\qquad = \left(a^{(0)} + a^{(1)}\alpha + \left(1 + a^{(1)}\right)\alpha^2\right)\left(\Gamma^{(0)} + \Gamma^{(1)}\left(\alpha + \alpha^2\right)\right). \end{aligned} \tag{16}$$

Expanding both sides and applying $\alpha^4 = \alpha + 1$ provides four equations:

$$\begin{pmatrix} b^{(0)} & b^{(1)} + 1 \\ b^{(1)} & b^{(0)} + b^{(1)} + 1 \\ b^{(1)} + 1 & b^{(0)} + b^{(1)} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \Delta^{(0)} \\ \Delta^{(1)} \end{pmatrix} = \begin{pmatrix} a^{(0)} & a^{(1)} + 1 \\ a^{(1)} & a^{(0)} + a^{(1)} + 1 \\ a^{(1)} + 1 & a^{(0)} + a^{(1)} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \Gamma^{(0)} \\ \Gamma^{(1)} \end{pmatrix}.$$

From the fourth line we immediately see $\Delta^{(1)} = \Gamma^{(1)}$. The second and the third line imply $\Delta^{(0)} + \Delta^{(1)} = \Gamma^{(0)} + \Gamma^{(1)}$ and hence $\Delta = \Gamma$. It follows that $A = B$ and this shows that the solution A is actually unique. Hence we can solve (15) to find the factor $g_j(Q) = a^{(0)} + a^{(1)}\alpha + \left(1 + a^{(1)}\right)\alpha^2$.

In [22] the authors only used one of the equations resulting from (14) and used the Weierstrass equation to determine $g_j(Q)$. In this way, they were not able to handle the general type of faults we consider here.

Recovery of the Secret From $g_j(Q)$. Once we know $g_j(Q)$ and therefore $g_j(Q)^{(1)}$ we can solve (13) for either (x_Q, y_Q) or (x_T, y_T), depending whether Q or P is the secret. If P is the secret, we can finally compute $P = \left(2^{-j} \mod l\right) T$ with $T = (x_T, y_T)$ (see also Remark 1).

Thus, we can completely recover the secret point in both cases when corrupting $g_j(Q)$ with faults of the type $\Delta^{(0)} + \Delta^{(1)}(\alpha + \alpha^2)$. This is due to a more careful analysis of (14) compared to [22].

5 Open Problems and Conclusion

For the most efficient implementations of the Tate pairing the first argument is defined over the base field and the second argument is defined over the extension field [2]. Extending the results of [21,11] we showed that it is in principle possible to attack the pairing no matter whether the secret is the first or the second argument of the pairing.

Several countermeasures like point blinding have been proposed to protect the pairing against SCAs [21,13,22]. They are all heuristic in the sense that they prevent a special attack and that their effectiveness is not rigorously proven. Hence, an important field of research is to find sound models that allow provable secure countermeasures that are efficient enough for the implementation on constraint devices like smart cards.

References

1. Acar, T., Lauter, K., Naehrig, M., Shumow, D.: Affine Pairings on ARM. IACR Cryptology ePrint Archive 2011, 243 (2011)
2. Barreto, P.S.L.M., Lynn, B., Scott, M.: On the Selection of Pairing-Friendly Groups. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 17–25. Springer, Heidelberg (2004)

3. Blake, I.F., Seroussi, G., Smart, N.P. (eds.): Advances in Elliptic Curve Cryptography. London Mathematical Society Lecture Note Series, vol. 317. Cambridge University Press (2005)
4. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. SIAM Journal on Computing 32(3), 586–615 (2003)
5. Hess, F., Smart, N.P., Vercauteren, F.: The Eta Pairing Revisited. IEEE Transactions on Information Theory 52(10), 4595–4602 (2006)
6. Joye, M., Neven, G. (eds.): Identity-Based Cryptography. In: Cryptology and Information Security, vol. 2. IOS Press (2009)
7. Joye, M., Tunstall, M. (eds.): Fault Attacks in Cryptography. Information Security and Cryptography. Springer (2012)
8. Kim, T., Takagi, T., Han, D.G., Kim, H., Lim, J.: Side Channel Attacks and Countermeasures on Pairing Based Cryptosystems over Binary Fields. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 168–181. Springer, Heidelberg (2006)
9. Miller, V.S.: The Weil Pairing, and Its Efficient Calculation. Journal of Cryptology 17(4), 235–261 (2004)
10. El Mrabet, N.: What about Vulnerability to a Fault Attack of the Miller's Algorithm During an Identity Based Protocol? In: Park, J.H., Chen, H.-H., Atiquzzaman, M., Lee, C., Kim, T.-h., Yeo, S.-S. (eds.) ISA 2009. LNCS, vol. 5576, pp. 122–134. Springer, Heidelberg (2009)
11. Mrabet, N.E., Flottes, M.L., Natale, G.D.: A practical Differential Power Analysis attack against the Miller algorithm. In: Research in Microelectronics and Electronics, pp. 308–311 (2009)
12. Page, D., Vercauteren, F.: Fault and Side-Channel Attacks on Pairing Based Cryptography. IACR Cryptology ePrint Archive 2004, 283 (2004)
13. Page, D., Vercauteren, F.: A Fault Attack on Pairing-Based Cryptography. IEEE Transactions on Computers 55(9), 1075–1080 (2006)
14. Barreto, P.S.L.M., Galbraith, S.D., Héigeartaigh, C.Ó.: Efficient Pairing Computation on Supersingular Abelian Varieties. Designes, Codes and Cryptography 42(3), 239–271 (2007)
15. Rosen, K.H. (ed.): Handbook of Elliptic and Hyperelliptic Curve Cryptography. Discrete Mathematics and its Applications. Chapman & Hall/CRC (2006)
16. Scott, M.: Computing the Tate Pairing. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 293–304. Springer, Heidelberg (2005)
17. Scott, M., Costigan, N., Abdulwahab, W.: Implementing Cryptographic Pairings on Smartcards. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 134–147. Springer, Heidelberg (2006)
18. Shoup, V.: A computational introduction to number theory and algebra. Cambridge University Press (2006)
19. Silverman, J.H.: The Arithmetic of Elliptic Curves, 2nd edn. Graduate Texts in Mathematics, vol. 106. Springer (2009)
20. Galbraith, S.D., Héigeartaigh, C.Ó., Sheedy, C.: Simplified Pairing Computation and Security Implications. Journal of Mathematical Cryptology 1(3), 267–281 (2007)
21. Whelan, C., Scott, M.: Side Channel Analysis of Practical Pairing Implementations: Which Path is More Secure? In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 99–114. Springer, Heidelberg (2006)
22. Whelan, C., Scott, M.: The Importance of the Final Exponentiation in Pairings When Considering Fault Attacks. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 225–246. Springer, Heidelberg (2007)