Improving side-channel attacks against pairing-based cryptography

Damien Jauvart, Nadia El Mrabet, Jacques A Fournier, Louis Goubin

Noname manuscript No.
(will be inserted by the editor)

Improving side-channel attacks against pairing-based cryptography

Damien Jauvart · Nadia El Mrabet · Jacques J.A. Fournier · Louis Goubin

Received: date / Accepted: date

Abstract Side-channel attacks are a serious threat against secret data involved in cryptographic calculations, as for instance pairing-based cryptography which is a promising tool for the IoT. We focus our work on Correlation Power Analysis (CPA) attack against a pairing implementation. We improve a vertical side-channel analysis attack and propose the first horizontal attack against a pairing implementation. First, we present a characterization of the multiplication that allows us to reduce by a factor of ten the number of side-channel traces required in order to perform a CPA attack against an implementation of Ate pairing. Secondly, we successfully attack the same implementation with only one trace by using the first horizontal attack path against pairing-based cryptography.

Keywords Side-channel attacks · pairing-based cryptography · collisions attack · countermeasure

Damien Jauvart
Aix-Marseille Université, 163 Avenue de Luminy, 13009 Marseille, France. E-mail: damien.jauvart@univ-amu.fr

Nadia El Mrabet
IMT, Mines Saint-Etienne, Centre CMP, Equipe Commune CEA Tech - Mines Saint-Etienne, F-13541 Gardanne FRANCE E-mail: nadia.el-mrabet@emse.fr

Jacques J.A. Fournier
Univ. Grenoble Alpes, CEA Leti, DSYS/LSOSP, Minatec Campus, 17 Rue des Martyrs, 38054 Grenoble Cédex, France. E-mail: jacques.fournier@cea.fr

Louis Goubin
Université de Versailles-St-Quentin-en-Yvelines, Laboratoire LMV, 45 avenue des Etats-Unis, 78035 Versailles Cedex, France. E-mail: Louis.Goubin@uvsq.fr

1 Introduction

The Internet of Things (IoT) is taking more and more importance in our daily life. It corresponds to numerous applications where the users' privacy and confidentiality must be guaranteed. In this IoT context, a major issue is the key generation. In a classical PKI model, the number of keys grows exponentially with the number of users [13]. Pairing-based cryptography provides elegant solutions for the key management, through Identity-based cryptography, and also allows very interesting protocols such as short signature schemes or hierarchical encryption [14]. We study the resistance of a pairing implementation against side-channel attacks in the context of IoT, namely an implementation of an Identity-based protocol on an embedded system. We try to recover the secret decryption key used in an Identity-based Encryption (IBE) scheme.

Side-channel attacks, which aim at recovering secret data, are a serious threat against cryptographic embedded devices. Indeed, in the case of embedded systems, the attacker can easily gain physical access to the device in order to perform this attacks [11,27,29].

Such attacks take advantage of physical information leakage during a computation. On a given embedded system, the leakage can be the execution time, the power consumption or the electromagnetic emission [27, 34]. Side-channel attacks use the link between leaked data and the processed values in order to retrieve information about the secret. Besides, we consider that the device is fully under the control of the attacker who can thus run the same computation with many known inputs.

As IBE [7] systems are not immune to these threats, the vulnerability of the pairing computations used in IBE systems should be investigated. In this case, the

decryption step is the target of the aforementioned side-channel attacks.

A pairing is a bilinear map denoted $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ are groups of prime order r. In practice $\mathbb{G}_1$ and $\mathbb{G}_2$ are sub-groups of an elliptic curve, $\mathbb{G}_3$ is a sub-group of a finite field. Let U be the ciphertext, the decryption step during an IBE protocol consists in evaluating $e(s, U)$ where $s \in \mathbb{G}_1$ is the secret key.

This critical operation was recently attacked through Correlation Power Analysis (CPA) [6,21,38] where the target was a modular multiplication calculation.

Studies on side-channel attacks share at least two important features: the comparison of side-channel leakage models [8] and the detection of points of interest related to the models [18]. The statistical tests that are used in order to detect points of interest can also be considered as a way of validating the leakage model.

Our approach gives rise to a parametrized attack. Because of the large number of variables, we provide a detailed characterization of how side-channel attacks leak information about critical operations during pairing computations. In order to illustrate the application of our approach in the context of a cryptographic algorithms, we targeted one of the modular multiplications involved in the software implementation of an Ate pairing, the secret to be recovered being one of the two points involved in this pairing calculation.

Compared to the practical attacks against pairing calculations published so far [6,38], our results, based on real electromagnetic measurements from the chip of an embedded 32-bit ARM core processor, require significantly less computational time in order to retrieve the secret value.

Our study proposes a generic method for attacking pairing implementations and defines parameters to increase a CPA's efficiency (namely the number of measurement curves needed). We use our method in order to develop, as far as we know, the first horizontal attack against a pairing implementation. Contrary to a classical vertical CPA that exploits several measurements at the same instant in time, a horizontal attack only uses one measurement curve at several instants. We illustrate the weakness of a pairing submitted to an horizontal attack and provide counter-measures in order to secure a pairing implementation. The horizontal approach is based on the fact that critical operations are performed several times during the same execution of the Miller algorithm. Measurements are selected at those moments to perform the attack. We also present the first electromagnetic attack against a protected pairing implementation.

The rest of the paper is organized as follows. Section 2 describes the mathematical background for pairings and an IBE pairing-based protocols. Section 3 contains a state of the art of the side-channel attacks against pairings. Section 4 describes improvements of the CPA attack against an unprotected pairing implementation. In Section 5 we provide an optimized side-channel attack, horizontal attack, against pairings with only one side-channel trace. Then, Section 6 shows different countermeasures from the literature and a collision attack to defeat one of them. Section 7 is devoted for a conclusion and prospect.

2 Cryptographic pairings

This section is divided into three parts: first the definition of a pairing, then the definition of the Ate pairing and the presentation of classical implementation tricks for pairings. The last part is a description of an Identity-based protocol: the Identity-based Encryption (IBE).

2.1 Elliptic curves and pairing definition

Let p be a large prime number. An elliptic curve E over the finite field $\mathbb{F}_p$ is defined by the reduced Weierstrass equation as follows:

$$E(\mathbb{F}_p) = \left\{(x, y) \in \mathbb{F}_p^2 | y^2 = x^3 + ax + b, \text{ for } a, b \in \mathbb{F}_p\right\} \cup \{\mathcal{O}\}, \tag{1}$$

where $\mathcal{O}$ is the point at infinity. The curve E should be smooth. This condition is satisfied if the discriminant $\Delta(E) = -16(4a^3 + 27b^2)$ is not zero.

Elliptic curves E are used in cryptography because we can define a group law on the set of its points. We can perform the addition of two different points of E and the doubling of a point of E.

For efficiency reasons, we can use a projective representation of the points of an elliptic curve. Indeed, a pairing computation is based on a scalar multiplication over an elliptic curve. In affine coordinates, a scalar multiplication involves several inversions, which are expensive operations. We describe our attack using the Jacobian coordinates as they were proved to allow the most efficient pairing computations [5]. Since then, it appears that projective coordinates provide the most efficient pairing implementations [1]. However, our methods and descriptions can easily be applied to any implementation of pairings using any system of coordinates. Indeed, the main ingredient of the attack is the interaction between secret data and public one during a multiplication over $\mathbb{F}_p$.

Among the points belonging to a curve E, we will work with points of order r (The order of a given point P is the smallest integer o such that $[o]P = \mathcal{O}$.). The set of points of order r is denoted by $E[r] = \{P \in E | [r]P = \mathcal{O}\}$. In practice we work on $E[r]$ with r a large prime dividing the cardinal of $E(\mathbb{F}_p)$. We know that $E[r] \subset E(\mathbb{F}_{p^k})$, for $k > 1$, where k is the embedding degree, *i.e.* the smallest integer such that r divides $p^k - 1$.

2.2 Ate pairing and implementation

For the description of our physical attacks against pairing implementations, it is not necessary to provide all the theoretical details about the Ate pairing and we refer the reader to [19] for a complete presentation. We only recall the definition of the Ate pairing. The sets $\mathbb{G}_1$ and $\mathbb{G}_2$ are sub-groups of order r of the elliptic curve $E(\mathbb{F}_{p^k})$, with r a large prime divisor of $\#E(\mathbb{F}_{p^k})$ the cardinal of $E(\mathbb{F}_{p^k})$. The set $\mathbb{G}_T$ is the multiplicative sub-group of the group of r-th roots of unity in $\mathbb{F}_{p^k}$.

$$\begin{aligned} \tau_r : \mathbb{G}_1 \times \mathbb{G}_2 &\to \mathbb{G}_T \\ (P, Q) &\mapsto (f_{t-1,Q}(P))^{\frac{p^k-1}{r}}, \end{aligned} \tag{2}$$

where $t = p+1-\#E(\mathbb{F}_p)$ is the trace of E. The function $f_{t-1,P}$ is called the Miller function.

Efficient implementations of such pairing computation are achieved through the efficient Miller's function computation and the final exponentiation.

2.2.1 Miller's algorithm

The goal is to build a function $f_{m,Q}$ which can be efficiently computed for the integer m. To solve this problem, a recursive method was described by Miller in [31], and is presented in Algorithm 1.

Algorithm 1: Miller's algorithm [31]

Input : An elliptic curve E over $\mathbb{F}_p$, $m = (m_{n-1} \dots m_0)_2$, $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.
Output : $f_{m,Q}(P)$.

$f \leftarrow 1$;
$T \leftarrow Q$;
for $i = n - 2$ ***to*** 0 **do**
  $f \leftarrow f^2 \cdot l_{T,T}(P)$ (tangent at T);
  $T \leftarrow [2]T$;
  if $m_i = 1$ **then**
    $f \leftarrow f \cdot l_{T,Q}(P)$ (line through T and Q);
    $T \leftarrow T + P$;
return f ;

Equations of line and tangent in Jacobian coordinates. For efficiency reasons, the points can be represented with mixed affine-Jacobian coordinates for the addition and doubling of points in the elliptic curves. The equation of the line through T (in Jacobian coordinates) and Q (in affine coordinates) evaluated at the point P (also in affine), $l_{T,Q}(P)$ is given by Equation 3.

$$\begin{aligned} l_{T,Q}(P) = &(y_P Z_T^3 - Y_T)(x_Q Z_T^2 - X_T) \\ &-(y_Q Z_T^3 - Y_T)(x_P Z_T^2 - X_T). \end{aligned} \tag{3}$$

The tangent equation at point T evaluated at P is provided by Equation 4.

$$\begin{aligned} l_{T,T}(P) = &2Y_T Z_T^3 y_P - 2Y_T^2 \\ &-(3X_T^2 + aZ_T^4)(x_P Z_T^2 - X_T). \end{aligned} \tag{4}$$

Those equations provide the targets for our attack.

2.2.2 Final exponentiation

The final exponentiation to the power $\frac{p^k-1}{r}$ is generally decomposed into three simpler exponentiations, when $k = 2\delta$:

$$\frac{p^k - 1}{r} = \left(p^\delta - 1\right) \frac{p^\delta + 1}{\Phi_k(p)} \frac{\Phi_k(p)}{r}. \tag{5}$$

where Φ_k refers to the k-th cyclotomic polynomial. Computations of the final exponentiation are widely described in [36]. In this paper we focus only on Miller's algorithm, because the weakness of a pairing implementation against side-channel attacks are localized in this step. Indeed, the secret is directly used during the Miller algorithm. We do not use the final exponentiation in our attack path in consequence of what we do not give any details about its implementation.

2.3 Identity-based Encryption

An Identity-based Encryption (IBE) scheme can be used to implement a widely known issue in public key cryptography: the key exchange protocol [37]. A Public-Key Infrastructure (PKI) based on IBE is scalable when compared to classical schemes using certificates, because precisely it does not require any certification [12]. The drawback is that the trusted authority must really be trusted and will not spy on the exchanges between Alice and Bob. For instance, the trusted authority can be the bank when Alice is a smart card and Bob a payment terminal.

In an IBE scheme, the public key is the user's identity. The associated private key is generated by the Private Key Generator (PKG) [7]. All Identity-based protocols share one common feature: at a given time there is a computation of a pairing where one of the parameter is secret.

As illustration, we summarize the Boneh-Franklin IBE scheme [7] which is composed of four steps:

1. Set-up;
2. Extraction;
3. Encryption;
4. Decryption.

During the Set-up step, the PKG generates the public parameters for the pairing:

$$\{r, n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, P_{PUB}, H_1, H_2\},$$

where H_1 and H_2 are two hash functions.

The extraction algorithm provides the user's private key denoted by d_B (computed using the identity of Bob and securely transmitted to him by the PKG).

During the encryption step, Alice only uses the public key of Bob. She sends him $U \in \mathbb{G}_1$. Bob's private key is involved during the decryption step in the pairing computation. Bob computes the pairing $e(U, d_B)$. Using this notation, the secret is the second input of the pairing. We will later discuss the influence of the secret position on the success of the attack.

The life cycle of the key When a Side-channel attack is performed, it is usually assumed that the secret is used more than once. We will consider this hypothesis for the classical CPA attack. However, in other scenarios[1], the key is used only once or a smaller number of times, so that the horizontal attack path becomes the only possibility for the attacker. We will show that horizontal CPA attacks are possible against a pairing implementation, even if the key is used only once.

3 Side-channel attacks against unprotected pairings implementations

In a pairing-based cryptographic protocol, the secret is one of the two points in the pairing's inputs. Attacks against the implementations of the pairing algorithms were first revealed by Page and Vercauteren [32].

[1] For instance if the secret involved in the pairing computation is a session key.

3.1 State of the art

Side-channel attacks against cryptographic algorithms have been widely studied for more than two decades [27]. Public-key cryptosystems such as RSA or ECC were demonstrated to be vulnerable when submitted to SPA and different variants of DPA. The objective is to reveal the secret exponent (RSA) or the secret scalar (ECC) used in a signature or decoding cryptosystem [10].

Several publications describe side-channel attacks against pairings over characteristics 2 or 3 fields. These studies are simply mentioned for reference, given that our implementation relies on large prime fields as described in Section 3.3. Page *et al.* [32] describe the principle of some physical attacks (passive side-channel and active attacks by injections of faults) on a pairing algorithm. They target the Duursma-Lee's [15] algorithm, which is used to calculate the Tate pairing on an elliptic curve over a finite field of characteristic 3. The manipulation of the data during the Duursma-Lee algorithm involves the product of a secret element with a value derived from the known input point of the pairing. The authors propose a SPA-like attack against modular multiplication algorithms which are implemented with the shift-and-add method. Furthermore, they describe a SPA attack which aims at getting the secret bit by bit. Kim *et al.* [25] propose temporal attacks, SPA and DPA to target the arithmetic operations which are involved in a pairing over curves in characteristic 2. In the context of the Eta pairing on binary fields, the targeted operation is $a(b + r)$, where a and b are derived from the secret, and r is derived from the known input. The authors conclude that, in theory, the DPA bit by bit proposed by Page *et al.* [32] could again recover the secret point used in the pairing computation. Pan and Marnane [33] propose a practical CPA attack based on a Hamming distance model against the Eta pairing over fields of characteristic 2 based on supersingular curves.

One of the first articles describing side-channel attacks against pairings of large characteristic field was proposed by Whelan *et al.* [39]. They describe a CPA targeting the arithmetic operations to recover the secret. For the calculation part of their CPA, they propose the following plan: calculations of the correlations between the hypothetical outputs of the arithmetic operation $x \times k$ for all the possible keys k and the leakage traces. For every key assumption, there is a correlation curve. The candidate key is the one providing the curve with the highest peak. In the same paper, the authors discuss the use of the words length (8, 16, 32 or 64) to represent multiprecision numbers. Indeed, they detail the computations of partial correlations which can be used by a CPA to target a part of the word. Especially,

if the target is a 32-bit architecture, the enumeration of sub-keys should be made on 2^{32} elements, which is considered as unreasonable today. So, it is necessary to target a part of these 32 bits, for example the least significant bytes, and thus to work with partial correlations. El Mrabet *et al.* [16] propose a simulation of the attack against the Miller algorithm of the pairing over the field $\mathbb{F}_{251}$. The equation of the tangent is targeted because it contains a modular multiplication of a coordinate derived from a public input point by a deterministic value derived from the secret point.

Ghosh *et al.* [17] detail a DPA against the modular subtraction in a Tate pairing implemented on a Barreto-Naehrig [3] elliptic curve. Blömer *et al.* [6] describe side-channel attacks against the modular additions and multiplications of elements in large prime characteristics. Just as El Mrabet *et al.* [16], they show that these attacks are possible even if the secret point is used as the first argument of the pairing computation. The authors illustrate their results on simulations. Indeed, they use simulation traces with a Hamming weigh model perturbated by a Gaussian noise. Unterluggauer and Wenger [38] publish the first practical attack against a pairing in large prime characteristic fields. They use a CPA-like approach, as previously described by Whelan *et al.* [39]. The authors target the modular operations during an Ate pairing in order to first find the secret 16 least significant bits and then the 16 most significant bits. They work on a 32-bit architecture and take advantage of the processor working with a 16-bit multiplier. Their configuration require more than 1500 traces to find the secret point. In 2016 the importance of implementing countermeasures is supported by the results of Jauvart *et al.* [21], where attacks are presented in a real world environment. Indeed, Ate pairings implemented on ARM Cortex-M3 have been broken efficiently with CPA attacks with approximatively 150 traces.

3.2 Details on classical attacks

We describe the attack principle against Miller's algorithm implemented with the Jacobian coordinates.

3.2.1 Case 1: Q is known and P is secret

The numerator of the tangent equation is $2Y_T Z_T^3 y_P - 2Y_T^2 - (3X_T^2 + aZ_T^4)(x_P Z_T^2 - X_T)$. The coordinates of the secret point P are involved in $2Y_T Z_T^3 y_P$ and $x_P Z_T^2$. These coordinates are attacked by a CPA against the modular multiplication. The coordinates of the point $T = (X_T : Y_T : Z_T)$ are known because they are initialized with those of the known point Q.

3.2.2 Case 2: P is known and Q is secret

In this case, the targeted equation is the same, we consider them at the first iteration of the Miller algorithm. Then, the coordinates of the point T are exactly the coordinates of the point Q, i.e. $T = (x_Q : y_Q : 1)$. Then the targeted tangent equation is $2Y_T y_P - 2Y_T^2 - (3X_T^2 + a)(x_P - X_T)$. The attacker is able to perform an attack against the modular multiplication $Y_T y_P$ in order to find $Y_T = y_P$. He then uses the equation of the elliptic curve to find x_Q. He finds several candidates x_1, x_2, x_3, because of the degree of the equation to be solved, he can test them until he finds the same result of the pairing together with the secret x_Q.

3.3 Target description

In order to implement some attacks in a real world environment, we use a real embedded device as target.

Pairing implementation under tests. We use our own software implementation of the Ate pairing over Barreto-Naehrig curves [3]. For such curves we recall that the parameters are p, r and the trace t defined with :

$$\begin{cases} p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1, \\ r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1, \\ t(x) = 6x^2 + 1. \end{cases} \tag{6}$$

We choose to use $x =$ `0x3FC0100000000000` given in hexadecimal form. The elliptic curve is $E : y^2 = x^3 + 5$. The tower field extension uses to build the field $\mathbb{F}_{p^{12}}$ is:

$$\begin{aligned} \mathbb{F}_{p^2} &\simeq \mathbb{F}_p[u]/(u^2 - \beta), \\ \mathbb{F}_{p^6} &\simeq \mathbb{F}_{p^2}[v]/(v^3 - u), \\ \mathbb{F}_{p^{12}} &\simeq \mathbb{F}_{p^6}[w]/(w^2 - v), \end{aligned} \tag{7}$$

where $\beta = -5$ is a non quadratic residue in $\mathbb{F}_p$, with u a non cubic residue in $\mathbb{F}_{p^2}$ and v a non quadratic residue in $\mathbb{F}_{p^6}$.

The representation of the points is performed via the following system coordinates:

- $P \in E(\mathbb{F}_p)$ is in affine coordinates.
- $Q' \in E_2(\mathbb{F}_{p^2})$ is in affine coordinates.
- $T \in E_2(\mathbb{F}_{p^2})$ is in Jacobian coordinates..

The computations for the doubling and addition of points over the elliptic curve are then performed with the tangent and line evaluation method, in what is called mixed coordinates.

Device under test. The target is a 32-bit microcontroller, implemented in 130 nm CMOS technology. This basic integrated circuit has no physical protections against Side-channela ttacks. This is an ARM Cortex-M3. The processor is clocked at 24 MHz. The sampling frequency of the oscilloscope is 4 giga samples per second. We target the modular multiplications. One multiplication takes 5 ms, then it is sampled over 20 million points.

Remark concerning recent attacks against the discrete logarithm problem. First, the curves implemented over binary and ternary fields are definitively to be avoided for cryptographic uses [23]. We have to use large prime characteristics. Recent constraints are published in [24, 28] and new curve parameters are recommended. In a few words, the size of the parameters (256 bits for p and r and our case) must be expanded to 460-bits at least [2] for BN curves. For the 128-bit security level, the curves providing the most efficient implementation seem to be the KSS16 family. Nevertheless, the algorithm implementation structure remains identical to our implementation. Mainly, the targeted operation for side-channel attacks, the modular multiplication, can still be computed with the CIOS multiprecision multiplication of Montgomery [26]. Even if another modular multiplication is implemented, our attack path remains successful. As a consequence, even if the parameters we use are no longer up to date, the success of our attack will be the same using new parameters.

3.4 Attacking the modular multiplication

The targeted modular multiplication $a \times b \mod p$ is the CIOS multiprecision multiplication of Montgomery [26]. It is often chosen for cryptographic applications, and finally, this choice does not affect the attack strategy. It is necessary to identify the operation which involves both a and b. In the case of the CIOS method, we have the calculation $(uv) \leftarrow c_j + \mathbf{a_j b_i} + u$. The choice of the algorithm for the modular multiplication (SOS,CIOS,FIOS,FIPS,CIHS [26]) does not change the target; it is still the multiplication of two machine words $a_j b_i$. The calculation of the intermediate variables, for example c_j and u must be adapted. We recall the CIOS method in Algorithm 2. We use the following notations: $\mathcal{W}$ is related to the architecture size, $\mathcal{W} = 2^{32}$ bits in our case. A long integer is represented by $\mathcal{N}$ words, in the case of 256-bits integer, we have $256 = \underbrace{32}_{\mathcal{W}} \times \underbrace{8}_{\mathcal{N}}$.

Algorithm 2: Montgomery Modular Multiplication CIOS [26]

Input : The modulus $p = (p_{\mathcal{N}-1} \dots p_0)_{\mathcal{W}}$ coprime to $\mathcal{W}$, $\mathcal{R} = \mathcal{W}^{\mathcal{N}}$, p' such that $\mathcal{R}\mathcal{R}^{-1} - pp' = 1$ and two integers $a = (a_{\mathcal{N}-1} \dots a_0)_{\mathcal{W}}$ and $b = (b_{\mathcal{N}-1} \dots b_0)_{\mathcal{W}}$.
Output : The integer $c = (c_{\mathcal{N}-1} \dots c_0)_{\mathcal{W}}$ such that $c = \left(ab\mathcal{R}^{-1}\right) \mod p$.

```
c ← 0 ;
for i = 0 to N − 1 do
    u ← 0 ;
    for j = 0 to N − 1 do
        (uv) ← c_j + a_j b_i + u ;
        c_j ← v ;
    (uv) ← c_N + u ;
    c_N ← v ;
    c_{N+1} ← u ;
    m ← c_0 p'_0 mod W ;
    (uv) ← c_0 + m p_0 ;
    for j = 1 to N − 1 do
        (uv) ← c_j + m p_j + u ;
        c_{j−1} ← v ;
    (uv) ← c_N + u ;
    c_{N−1} ← v ;
    c_N ← c_{N+1} + u ;
if (c_{N−1} ... c_0)_W < p then
    c ← (c_{N−1} ... c_0)_W ;
else
    c ← (c_{N−1} ... c_0)_W − p ;
return c ;
```

3.5 Attacks against 32-bit multiplications

There are two distinct cases to be studied:

- Either b is known and a is secret.
- or a is known and b is secret.

Case 1: b is known and a is secret. The first iteration of the CIOS algorithm corresponds to the indexation $i = j = 0$. The targeted instruction is $(uv) \leftarrow a_0 b_0$, the multiplication of two machine words is already attacked in [6,16,38,39]. It is a CPA where the target is the result of the multiplication of machine words. The predictions of the multiplication between the inputs (one seen as a secret key, the other as a known data) are correlated with the measurements traces. The attack allows to recover a_0. The targeted word is now a_1, it is manipulated for $i = 0$ and $j = 1$, the targeted instructions $(uv) \leftarrow a_1 b_0 + u$, where u is the carry of the previous operation, *i.e.* $(uv) \leftarrow a_0 b_0$, thus u is known. It is then possible to use a CPA in order to find a_1. Other words of $a = (a_{n-1} \dots a_0)_{2^{32}}$ are recovered in the same way. We have to consider the carry as the information leakage is due to access to memories and more particularly the write-back on the memory.

Case 2: a is known and b is secret. This case targets the words b_i, it requires a more thorough study of the algorithm because the index i is the one of the main loop. The first iteration ($i = j = 0$) produces the operation $(uv) \leftarrow a_0 b_0$, which allows to find b_0 as previously described. In order to identify the carry u through the iterations, we denote it by u_{00}.

The target is the word b_1. It is involved in the instruction $(uv) \leftarrow c_0 + a_0 b_1$. The word a_0 is known, we are still left to see if we can predict c_0.

This word c_0 comes from the instruction $c_{j-1} \leftarrow v$ of the CIOS algorithm for $i = 0$ and $j = 1$. At first, there is the calculation $(uv) \leftarrow c_1 + mp_1 + u$, afterwards, the instruction $c_0 \leftarrow v$. Then we have to verify that the words c_1, m and u are known (p_1 is obviously known since it depends on the modulo p). The Table 1 shows the dependence between the intermediate variables and how we reach in c_1, m and u.

Table 1: Origins of the intermediate variables to calculate c_0

Variables	**Origins**		
	Operations	**Intermediate variables**	
		Known	**Unknown**
c_1	$a_1 b_0 + u_{00}$	a_1, b_0, u_{00}	$-$
u	$c_0 + mp_0$	$-$	c_0, m
m	$c_0 p'_0 \mod 2^{32}$	p'_0	c_0
c_0	$a_0 b_0$	a_0, b_0	$-$

4 Reducing the number of traces for CPA against pairings

The state of the art presented below shows how the classical attacks can be used to recover the secret input of a pairing function. However few practical validations of these attacks have been performed up to now. In this section we present the most effective known attack [21] and describe the advanced techniques we use to make it powerful. Our method improves the efficiency of the attack as concerns the number of traces (factor 10) and the necessary resources.

4.1 Attack scheme

In order to recover the 32 bit of a secret operand κ involved in multiplications $x^{(i)} \times \kappa$ for known κ, we proceed with a divide-and-conquer strategy.

In a first part, we describe the attack path on a word of size $2z$. We first attack the z least significant bits, and then the z most significant bits. Figure 1 illustrates this principle. The method is the following:

1. We enumerate the 2^z key hypotheses on z bits (this set is denoted by $\mathcal{K}_0$) and apply a CPA attack: for each hypothesis, we compute the intermediate variable (for each plaintext) and the correlation between each predicted intermediate variable and the side-channel observation. We keep the α best key hypotheses which maximize the correlation. The set of hypotheses is denoted by $\mathcal{K}_1 = \{\kappa_{1,1}, \ldots, \kappa_{1,\alpha}\}$.
2. We enumerate the 2^z most significant bits from $\kappa_{1,i} \in \mathcal{K}_1$ hypothesis, and we apply a CPA. For each $i = 1, \ldots, \alpha$, we compute the 2^z key hypotheses: $\kappa_{2,i,j} = \underbrace{(j_7, \ldots, j_0)_2}_{j=0,\ldots,2^z-1 \text{ radix } 2} \| \kappa_{1,i}$.

   The hypotheses of $\mathcal{L}_i = \{\kappa_{2,i,0}, \ldots, \kappa_{2,i,2^z-1}\}$ are on $2z$ bits: for each of them, we compute the correlation with the measured traces. Thereafter, for $i = 1, \ldots, \alpha$ we hold the α best key candidates and store them in $\mathcal{L}'_i$.
3. The key in $\cup_{i=1}^{\alpha} \mathcal{L}'_i$ with the best correlation is the key candidate for the $2z$ secret bits.

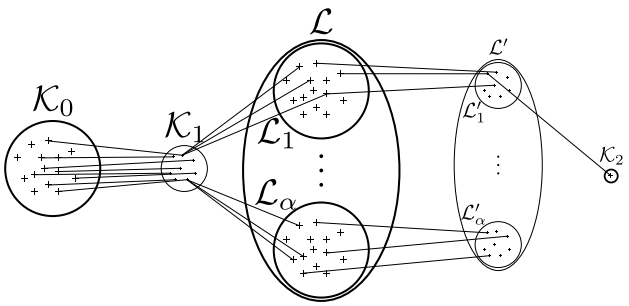


Fig. 1: Attack scheme on $2z$ bits

4.2 Practical attacks

First, we compute a CPA-based characterization on 32 bits. Then, we compare two Hamming weight leakage models on 8 bits, and we finish by the CPA attack with the best models against the 32 secret bits. The choice of the model on 8 bits was determined by an exhaustive research. We also tried $z = 4, 6, 12$ but the attack were not successful.

4.2.1 CPA characterization on 32 bits

For this characterization step, some key hypotheses are randomly chosen on 32 bits, knowing that the secret

key is among them. To predict an intermediate step we compute $\phi(\kappa, x)$ the Hamming weight of x and κ, where x is the plaintext and κ is the key hypothesis, the result is a 32-bit integer.

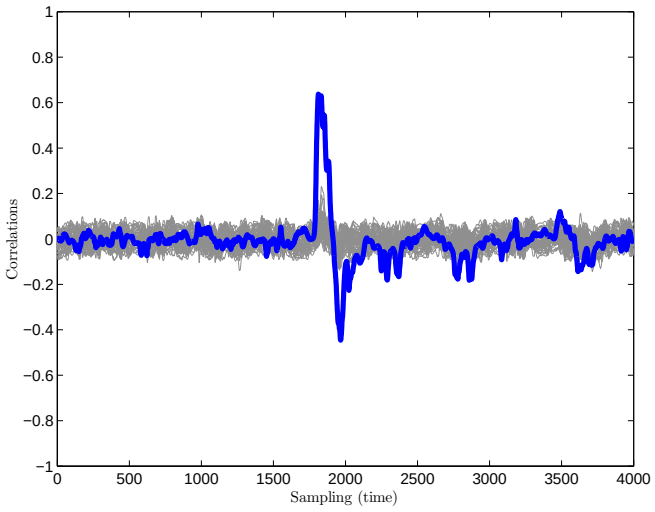


Fig. 2: Characterisation CPA on 32 bits

Among the different key hypotheses, the correct secret key has the highest correlation. Figure 2 shows the correlation of the good key in bold, it is clearly higher than the other ones.

4.2.2 Hamming weight models

In practice, since it is not possible to enumerate the 2^{32} key hypotheses, we start with the $z = 8$ least significant bits. Then, κ is on z bits, the plaintext can be taken on 32 bits, so the product result is either on z bits or on $2z$ bits as shown in Figure 3.

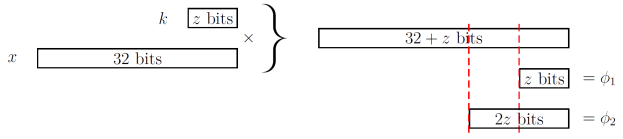


Fig. 3: Leakage models

We estimate both leakage models previously introduced by means of the t-test. This statistical test was introduced for the detection of points of interest by Gierlichs *et al.* [18] under the name of SOST (Sum Of Squared pairwise T-differences). For that purpose, we use the previous measurements. The key κ is fixed and the messages are changed. We fix $z = 8$ bits. For each trace, the calculation of $\phi(\kappa, x)$ returns a set. In the end, the size of every set is denoted $\eta_{\phi,i}, i = 1, \ldots, N_\phi$. In the case of the first model there are $N_{\phi_1} = 9$ sets (9 possible Hamming weights for 8 bits). And for the second model $N_{\phi_2} = 17$. For each set, the average m_{ϕ_i} and the variance $\sigma^2_{\phi_i}$ are computed. The SOST is then obtained using Equation 8:

$$SOST_\phi = \sum_{i,j=1}^{N_\phi} \left(\frac{m_{\phi,i} - m_{\phi,j}}{\sqrt{\frac{\sigma^2_{\phi,i}}{\eta_{\phi,i}} + \frac{\sigma^2_{\phi,j}}{\eta_{\phi,j}}}} \right)^2 \quad \text{for } i \geq j. \tag{8}$$

The results of the t-test for both models are provided in Figure 4.

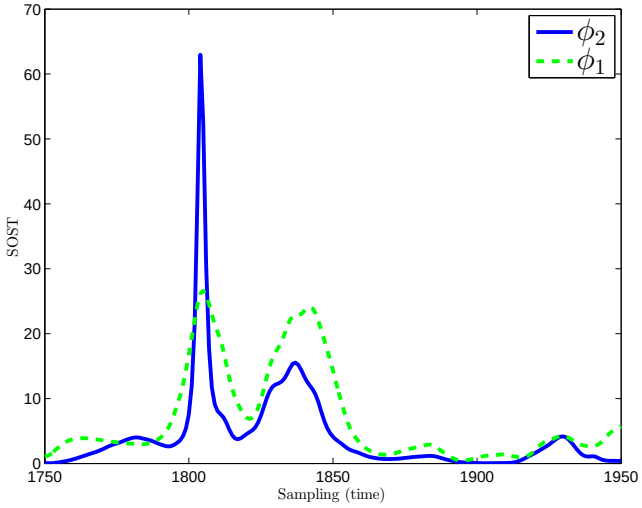


Fig. 4: T-test for both Hamming weight models

For this figure, a zoom is made on the part containing the leakage. The leakage is further highlighted when the second model is used: the peak is more narrower and also higher.

4.2.3 Practical results

We have to analyze the number α of key candidates after each intermediate attack. Unterluggauer and Wenger [38] only tested the values $\alpha = 5$ and $\alpha = 10$, which did not show any differences. However, we demonstrate that for a fixed number of traces used for the attack, the success rate increases with α [21]. In Figure 5, for example, for a database of 80 traces, with $\alpha \geq 40$ the success rate of the attack is above 90%. For 170 traces, the attack has a success rate of 100% as soon as $\alpha \geq 28$.

The attack against an operand of the multiplication of machine words is efficient whatever the position of

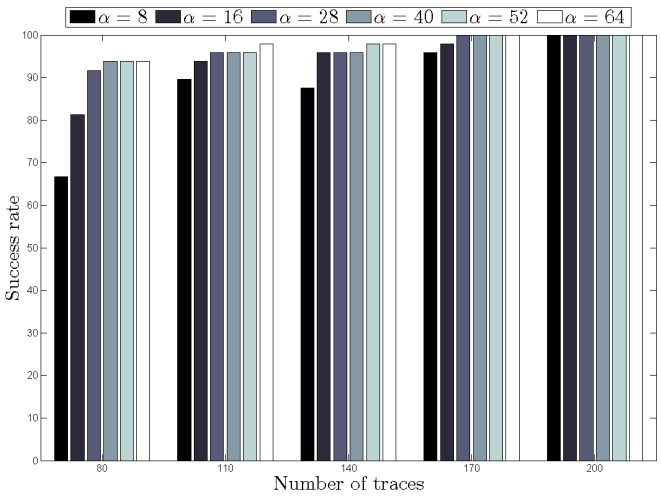


Fig. 5: Attacks results for some values of α

the secret. With the results of Section 3.5 we are able to attack a modular multiplication, in particular when it is implemented in CIOS. Since the attack against pairing targets the modular multiplication, this attack is able to effectively recover the secret pairing input point.

Table 2 shows the differences when recovering a 32-bits word. "Time" denotes the number of enumerated keys and "Memory" is the number of keys which are temporarily stored.

Table 2: Attacks comparisons: resources and required traces

	[38]	**Our**
Time	2^{19}	2^{16}
Memory	2^{19}	2^{16}
Number of trace	1500	**150**

5 Horizontal attack against pairings

As presented in Section 3, a classical side-channel attack exploits several traces at the same time. The points of interest are the window corresponding to the targeted operation processed by the integrated circuit. The horizontal approach is the following: a critical operation is performed several times within the same execution of the algorithm. The targeted window is repeated, the traces are then selected at these times in order to make a CPA-like attack (for instance) [4,9].

The structure of the Miller algorithm is asymmetric in P and Q. More precisely, the temporary point T, which is allocated to the value of Q at the beginning of Miller's algorithm, is going to evolve during the iterations. The study of this structure allows us to identify a weakness.

This is the first horizontal side-channel attack against a pairing implementation.

5.1 Weakness in the Miller algorithm

The computation of the tangent equation, in mixed affine-Jacobian coordinates (Equation 4), involves a multiplication between x_P and Z_T^2. An execution of the Miller algorithm comprises $n-1$ iterations, thus $n-1$ calculations of tangent, and thus $n-1$ multiplications $x_P \times Z_T^2$ with a fixed x_P and different Z_T.

In case the point P is the secret, each execution of a pairing allows to observe $n-1$ multiplications which involve x_P and a known integer. The attack becomes a classic CPA with $n-1$ traces.

This weakness only appears when the point P is secret (notation for an Ate pairing $e(P,Q)$).

If the secret is the point Q, this point is modified at each step of the Miller algorithm. The only operations involving the coordinates of Q occurs during the addition step. Roughly, the number of addition steps is half the number of doubling steps. We will first describe the horizontal attack in the most favorable set-up, i.e. when P is secret. We will later discuss the case where Q is secret.

5.2 Practical validation of the horizontal attack

Our target is an Ate pairing, whose structure is similar to the Tate pairing. It is an optimized version, which uses twice less iterations for the Miller loop. The number of iterations is 126 for the used Barreto-Naehrig curves [3], we thus obtain 126 traces of Montgomery's modular multiplications $a \times b \mod p$ with b being the secret.

As illustrated in Figure 5 we can see that a small number of traces is sufficient for a successful attack. Namely the number of traces can easily been chosen under 126.

The attack context is the second case of Section 3.5. We choose a small value for α in the first place: $\alpha = 8$. This small value is among the ones which requires the smallest resources to make the attack successful. As the attack with $\alpha = 8$ allows to find the secret b, it is thus useless to make other attacks with $\alpha > 8$.

Figures 6 to 9 show the results of the attacks for the secret word b_0 of $b = (b_{N-1} \dots b_0)_{2^{32}}$. More precisely, Figure 6 gives the results of the correlations af-

ter the first sub-attacks. The bold thick curves represent the correlations for the correct secret sub-key on 8 bits $(b_{0,7} \dots b_{0,0})_2$: it is among the best hypotheses. Figure 7 represents the correlations after the second sub-attacks. Then the following attack is exposed in Figure 8. Finally the last sub-attacks are represented by Figure 9.

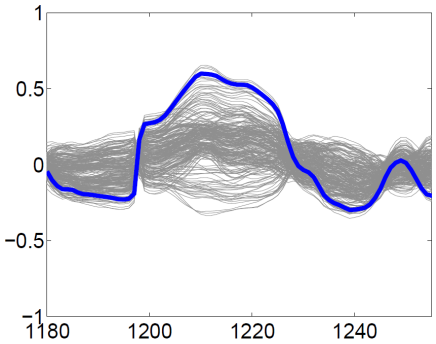


Fig. 6: Correlations for key candidates for the byte 1

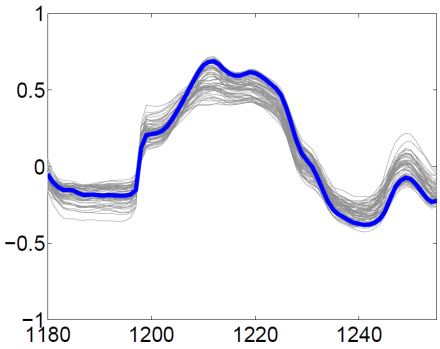


Fig. 7: Correlations for key candidates for the byte 2

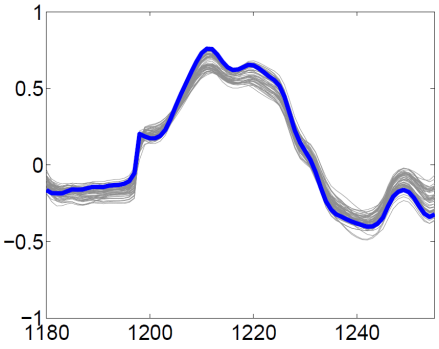


Fig. 8: Correlations for key candidates for the byte 3

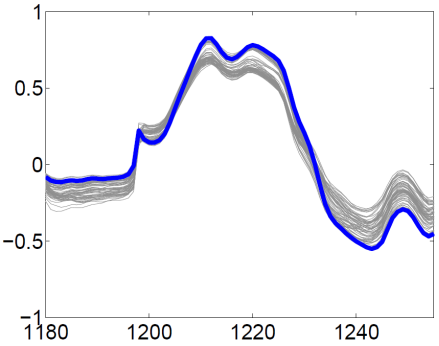


Fig. 9: Correlations for key candidates for the byte 4

In the last sub-attack (Figure 9) the bold thick curve has the highest amplitude, this provides the corresponding key hypothesis returned by the attack. Thus the attack allows us to recover the secret word b_0. The results are similar during the attack against the other words $b_{N-1}, \dots, b_1$. Our horizontal attack recovers the entire secret key using only one Ate pairing execution.

5.3 Discussion

In the presented attack carried in a real target we recover the secret integer (256 bits) with just one Ate-pairing execution. This is due to the use of our horizontal attack. This attack exploits the Miller loop's structure composed of 126 iterations. In the case of an Optimal-Ate pairing, the number of iterations is lower (which makes the algorithm more efficient). Let us see whether this optimization prevents our horizontal attack. The Optimal-Ate pairing over BN curves with the same security parameters gives a loop of at least $\frac{\log_2(r)}{\varphi(k)} = \frac{256}{4} = 64$ iterations, where φ is the Euler totient function. Doing a CPA with only 64 traces is more difficult.

However, recent attacks on the discrete logarithm [24, 28] indicate that the lengths of the parameters need to be increased. Then, for BN curves, the bitlength of r is now 460 bits instead of 256. The number of iterations in the Miller algorithm to compute the Optimal-Ate pairing is then $\frac{460}{4} = 115$. By putting this value in Figure 5 we see that our CPA attack when P is secret, for $\alpha = 64$, gives a success rate of 97% for 110 traces. This suggests that it is always possible to make such an horizontal attack. According to the analysis made by the authors in [24], for a 128-bit security level, the curve providing the most efficient implementation should be the KSS16 family. The optimal Ate over this family is performed in 77 iterations. It appears from Figure 5 that with 77 iterations of the Miller algorithm, the success rate of our attack is at most 90%. As a consequence,

in order to be sure to recover the correct key, the easy way would be to perform a horizontal attack using 2 measurement curves. Alternatively, we can try to obtain partial results for the key, and use a brute force approach for the remaining 20 bits.

If the secret is the point Q, we only have $\phi(\lambda)$ slots for performing the CPA attack, where λ represents the number of iterations during the Miller algorithm and ϕ the Hamming weight function. If the key is ephemeral and used η times, the attack would be successful if $\eta \times \lambda$ is greater than 110.

6 Attack against protected implementation

We saw in Sections 4 and 5 that unprotected pairings do not resist to side-channel attacks. First, we illustrated how to reduce the number of traces needed to successfully complete an attack, and then we showed the efficiency of a horizontal attack scheme. It is therefore more important than ever to protect such implementations. We recall existing countermeasures and demonstrate that some of them are not effective. We present here the first real attack by electromagnetic observations against a protected pairing implementation.

6.1 Guide of existing countermeasures

W can find in the literature two categories of countermeasures for pairing implementations: external or internal ones.

6.1.1 External countermeasures

The first countermeasure is proposed by the authors of the first attack. Page and Vercauteren [32] suggest to exploit the bilinearity property of the pairings. If a and b are integers coprime to r, then $e([a]P, [b]Q)^{1/ab} = e(P, Q)$. The countermeasure uses this randomization so that the attacker loses the knowledge of the coordinates of the public point. The calculation of $e(P, Q)$ is replaced by Algorithm 3. We shall name this countermeasure "multiplicative masking". We can omit the exponentiation at the power $1/(ab)$ if we choose a and b such that $ab = 1 \mod (r)$.
The second countermeasure of Page and Vercauteren [32] consists in using an additive mask. For example, if Q is the secret, then the point P will be masked as follows: $e(P, Q) = e(P + R, Q)e(R, Q)^{-1}$ or $e(P, Q) = e(P + R, Q)e(-R, Q)$ for a random point $R \in \mathbb{G}_1 \setminus \{\mathcal{O}, -P\}$. Algorithm 4 describes the principle of this countermeasure by additive masking.

Algorithm 3: First countermeasure of Page *et al.* [32]

Input : $P \in \mathbb{G}_1$ et $Q \in \mathbb{G}_2$.
Output : $e(P, Q)$.

1 a and b are two integers such that $ab = 1 \mod r$;
2 $P' \leftarrow [a]P$; $Q' \leftarrow [b]Q$;
3 $h \leftarrow e(P', Q')$;
4 **return** h ;

Algorithm 4: Second countermeasure of Page *et al.* [32]

Input : The secret point $Q \in \mathbb{G}_2$ and $P \in \mathbb{G}_1$.
Output: $e(P, Q)$.

1 $R \in \mathbb{G}_1 \setminus \{\mathcal{O}\}$ is randomly chosen ;
2 $P' \leftarrow P + R$;
3 $h_1 \leftarrow e(P', Q)$;
4 $h_2 \leftarrow e(-R, Q)$;
5 $h \leftarrow h_1 h_2$;
6 **return** h ;

6.1.2 Internal countermeasures

Kim *et al.* [25] proposed the use of the third countermeasure of Coron [10]. Indeed, the aim is to take advantage of the representation of points in projective coordinates. Let us assume that the Miller algorithm is implemented in projective coordinates. Instruction 2 of Algorithm 1 initializes the point T with the coordinates of P as follows:

$$X_T \leftarrow x_P; \quad Y_T \leftarrow y_P \text{ and } Z_T \leftarrow 1. \tag{9}$$

This initialization is replaced by:

$$\begin{array}{l} i.\ \lambda \text{ is randomly choosen in } \mathbb{F}_p^\star; \\ ii.\ X_T \leftarrow \lambda x_P; \quad Y_T \leftarrow \lambda y_P \text{ and } Z_T \leftarrow \lambda. \end{array} \tag{10}$$

In this cas, the output of the pairing calculation is equal to the output without randomization, because the final exponentiation maps all the elements of $\mathbb{F}_p^\star$ to 1, which applies to the mask λ. This countermeasure, based on the randomization of the coordinates, is less expensive: it requires only two additional multiplications in $\mathbb{F}_p$. The protection method proposed by Scott [35] is rather similar because it is based on the idea that it is possible to multiply by an element of $\mathbb{F}_p$ without adjustment on the input. It suggests masking the sensitive calculations, that is the multiplications between a known and a secret operand, by applying the multiplicative randomization by $\lambda \in \mathbb{F}_p^\star$.

This countermeasure is applied to randomize the public point P. This operation of randomization appears during the initialization of the temporary point

T with P. A weakness appears at this moment. In Jacobian representation, the coordinates of a given point are not unique, indeed they satisfy the equation

$$(x, y) = \left(XZ^2 : YZ^3 : Z\right), \forall Z \neq 0. \quad (11)$$

The principle of the countermeasure is to make the affectation of T with one $\lambda \in \mathbb{F}_p^\star$ different at each execution. The resulting Miller algorithm with this countermeasure is described by Algorithm 5.

Algorithm 5: Miller algorithm with randomisation of mixed coordinate affine-jacobian (Q is secret)

Input : An elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$, $m = (m_{n-1} \dots m_0)_2$, $P \in E\left(\mathbb{F}_{p^k}\right)[m]$ et $Q \in E\left(\mathbb{F}_{p^k}\right)$.
Output : $f_{m,P}(Q)$ with $\mathrm{div}(f_{m,P}) = m(P) - (mP) - (m-1)(\mathcal{O})$.

1 $f \leftarrow 1$;
2 $\lambda \in \mathbb{F}_p^\star$ is randomly chosen ;
3 $T = (X_T : Y_T : Z_T)$ with $X_T \leftarrow x_P\lambda^2$; $Y_T \leftarrow y_P\lambda^3$; $Z_T \leftarrow \lambda$;
4 **for** $i = n - 2$ **to 0 do**
5 $H \leftarrow 4X_TY_T^2$; $I \leftarrow 3X_T^2 + aZ_T^4$;
6 $X_T \leftarrow -2H + I^2$;
7 $Y_T \leftarrow -8Y_T^2 + I(H - X_T)$;
8 $Z_T \leftarrow 2Y_TZ_T$; `// Updating T (Doubling)`
9 $l_{T,T}(Q) \leftarrow 2Y_TZ_T^3y_Q - 2Y_T^2 - I(x_QZ_T^2 - X_T)$; `// Tangent evaluation`
10 $f \leftarrow f^2 l_{T,T}(Q)$;
11 **if** $m_i = 1$ **then**
12 $A \leftarrow y_PZ_T^3 - Y_T$; $B \leftarrow x_PZ_T^2 - X_T$;
13 $X_T \leftarrow A^2 - B^2(X_T + x_PZ_T^2)$;
14 $Y_T \leftarrow B^2(A(X_T - X_TZ_T^2) - BY_T)$;
15 $Z_T \leftarrow Z_TB$;
16 ; `// Updating T (Addition)`
17 $l_{T,P}(Q) \leftarrow B(Z_T^3y_Q - Y_T) - A(Z_T^2x_Q - X_T)$; `// Line evaluation`
18 $f \leftarrow fl_{T,P}(Q)$;
19 **return** f ;

6.2 Provided protection with state of art countermeasures

The target of a vertical CPA attack is the modular multiplications between coordinates depending on the secret Q and known integers. In Algorithm 5, the operations which involve x_Q and y_Q are the following:

- $2Y_TZ_T^3y_Q = h(T) \times y_Q$, where h depends only on T.
- $x_Q \times Z_T^2$.
- $Z_T^3 \times y_Q$.
- $Z_T^2 \times x_Q$.

All these multiplications are made between coordinates of Q and for random (unpredictable) integers. Indeed, the coordinates of T are not known and the CPA cannot be applied any more to find the secret Q because the predictions on the intermediate variables are no longer possible. Indeed, $\lambda \in \mathbb{F}_p^\star$ takes a random value in $\{1, \dots, p-1\}$ thus λ^2 has $\frac{p-1}{2}$ possible values, which implies that $X_T \leftarrow x_T\lambda^2$ is an unpredictable value among a set of cardinal $\frac{p-1}{2}$. The values of X_T, Y_T and Z_T being unpredictable, it is not possible any more to predict the internal states of the algorithm, and thus to achieve the CPA.

6.3 Aimed target

In the scenario of the IBE, the decryption of the ciphertext $\{U, V\}$ by Bob involves the step $K = e(U, d_B)$ where d_B is the secret key of Bob. If Bob loses the device where $e(U, d_B)$ is executed, then an attacker can control the input U which is manipulated. The flaw that we present here comes from paper [22] which describes in details the technical aspects.

The calculation $K = e(U, d_B)$ is computed with the Miller algorithm, where the countermeasure using coordinates randomization is used as presented in Algorithm 5.

Let us linger over the randomization operations and the calculation of the tangent in the first iteration. The randomization of the known point P is made in line 3, which involves the operation $X_T \leftarrow x_P\lambda^2$. In the computation of the tangent in the first iteration, the calculation $x_QZ_T^2$ is performed with $Z_T = \lambda$. Both executed operations are multiplications between a known coordinate (x_P) with a mask $\lambda' = \lambda^2$, then a secret coordinate (x_Q) with the same mask. In the following the notation λ replaces λ' for more readability.

As a consequence, if $x_P = x_Q$ then the same calculation is executed twice. If the side-channel allows to identify such a situation thanks to a technique of collision detection, the attacker is able to recover the secret x_Q. This value is the same as the coordinate x_P, which is known.

If the values of x_P and x_Q are equal (or partially equal) then the measured EM/power traces are similar. The data x_P and x_Q are multiprecision integers, for example 8 words of 32 bits. Thus it is impossible to test 2^{256} values for x_P. However, the device computes the modular multiplication by handling the words individually and in a deterministic way. For example, Montgomery's modular multiplication occurs at a precise instant in time, to do the multiplication $x_0\lambda_0$ (here we denote $x = x_P$ to ease the reading). Even with this re-

mark, the attack would consist in making an exhaustive search on 32 bits. As for the CPA, the attack against 32 bits is made byte by byte. That's why the words x_0 and k_0 are said to be partially equal if they have their least significant bytes identical (k corresponds to the secret coordinate, that is x_Q).

The remaining part of this section is devoted to the detection of similarities between two words x_0 and k_0 during their manipulation in a multiplication with the same unknown integer λ.

6.4 Attack schemes against a protected implementation

In order to exploit the above attack, the naive approach for correlation computations can be considered. Let $c^{(i,j)}$ denotes the correlation coefficient between traces $C_x^{(i,j)}$ and $C_k^{(i,j)}$, corresponding respectively to the operations $x_0^{(i)}\lambda_0^{(j)}$ and $k_0\lambda_0^{(j)}$. With this tool, a sketch of the attack is given by Algorithm 6, where N is the number of repetitions which the attacker can make with the same message $x_0^{(i)}$. After having kept N measurements, the cross correlation coefficients are calculated, then averaged. Concretely, taking the average of the correlations rather than the traces is useful to give a value of the real similarity between the operations $x_0^{(i)}\lambda_0^{(j)}$ and $k_0\lambda_0^{(j)}$ without taking into account the effect of the mask.

Algorithm 6: Naive attack scheme

Input : The plaintexts $x_0^{(i)}$, a positive integer N.
Output : A key candidate $\widehat{k}$ for the 8 least significant bits of k.

$c \leftarrow \mathcal{M}_{1,255}(0)$;
for $i = 0$ **to** 255 **do**
  for $j = 0$ **to** N **do**
    $C_x^{(i,j)} \leftarrow$ trace of the operation $x_0^{(i)}\lambda_0^{(j)}$;
    $C_k^{(i,j)} \leftarrow$ trace of the operation $k_0\lambda_0^{(j)}$;
    $c^{(i,j)} \leftarrow \rho\left(C_x^{(i,j)}, C_k^{(i,j)}\right)$;
  $c(i) \leftarrow \frac{1}{N}\sum_j c^{(i,j)}$;
$\widehat{k} \leftarrow \arg\max |c|$;
return $\widehat{k}$;

In [22] the authors show that the naive correlation search scheme does not result in a successful attack.

6.4.1 Brief description of vertical detection

In the horizontal approach, traces are compared two by two with a coefficient of similarity before being averaged. The vertical approach consists in comparing the database directly, *i.e.* without computing the average of the correlation coefficients. For that purpose, a temporal point taken in traces is denoted by $\left\{C_x^{(i,j)}\right\}_j$, these data are put in a vector of dimension N. Then, we also indicate a point in traces $\left\{C_k^{(i,j)}\right\}_j$, these data are also put in a vector. Finally, the correlation coefficients for these two vectors are calculated. Figure 10 illustrates this principle.

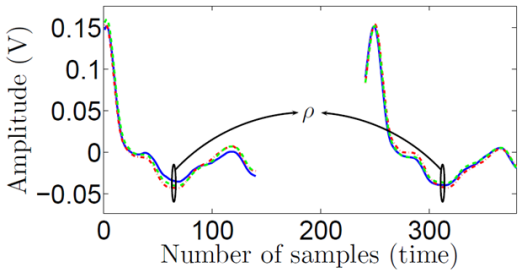


Fig. 10: Illustration of the vertical collision detection

The attacker is then able to create an output correlation for each of the 256 key hypotheses as illustrated by Equation 12:

$$c_i = \rho\left(\begin{pmatrix} C_{x,m_{\text{interest}}}^{(i,1)} \\ \vdots \\ C_{x,m_{\text{interest}}}^{(i,N)} \end{pmatrix}, \begin{pmatrix} C_{k,m}^{(i,1)} \\ \vdots \\ C_{k,m}^{(i,N)} \end{pmatrix}\right), \forall m \in I, \qquad (12)$$

for $i = 0 \ldots 255$. The attack ends by sending back the key candidate which maximizes the correlations, that is $\widehat{k} \leftarrow \arg\max_i |c_i|$.

6.4.2 Attack result against pairings

Figure 11 shows the correlations c_i for a vertical correlation collision detection attack by using $N = 400$ repetitions. The thickest curve corresponds to the correct sub-key hypothesis. We see on Figure 11 that the attack allows to distinguish the good sub-key hypothesis. Indeed, the bold thick curve, corresponding to the good sub-key hypothesis (on 8 bits) has a higher correlation than the other hypotheses.

Because the attack works with $N = 400$ repetitions (*i.e.* $400 \times 256 = 102400$ total number of traces), we modify this parameter in order to see the evolution of the rank for the good sub-key. For $N \in \{100, 110, \ldots, 400\}$, Figure 12a shows this evolution.

From $N = 350$, the attack allows to distinguish the good sub-key (the least significant byte of k_0).

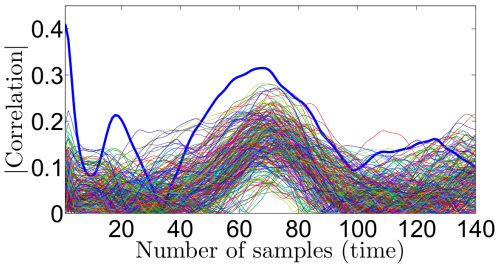
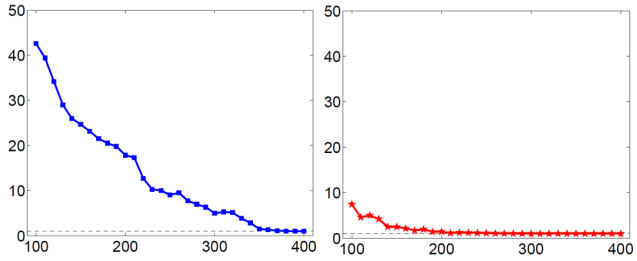


Fig. 11: Correlations related to a vertical correlation collision detection attack

When the least significant bits are recovered, the same attack can be applied to the next byte. Collision detection has thus been made easier, Figure 12b shows the evolution of the rank of the good key according to the number N of repetitions. It is enough to take approximately $N = 250$ to be able to find the 8 targeted bits.



(a) Byte 1 (b) Byte 2

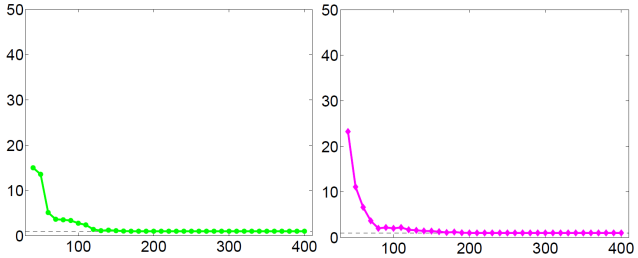
Fig. 12: Evolution of the rank of the good key for the attack with vertical correlation: byte 1 and 2

We follow the same approach for the successive bytes. This procedure gives even better results for byte #3. More precisely, Figure 13a shows that with $N = 200$ traces, the attack allows to find the correct key byte[2]. Finally, when the target is the last byte, Figure 13b shows that $N = 200$ is enough to identify 8 bits of the secret.

In view of the results given in Figures 12 and 13, the number of curves needed to find a 32-bit word is:

$$E_{k_0} \simeq 2^8 \left(\underbrace{400}_{\text{Byte 1}} + \underbrace{250}_{\text{Byte 2}} + \underbrace{200}_{\text{Byte 3}} + \underbrace{200}_{\text{Byte 4}} \right) \simeq 2,7 \times 10^5. \quad (13)$$

[2] We improve the efficiency of the attack when compared to [22] where 300 curves were necessary to perform the attack



(a) Byte 3 (b) Byte 4

Fig. 13: Evolution of the rank of the good key for the attack with vertical correlation: byte 3 and 4

Given that there are 7 other 32-bit words to be found, the total number of traces is:

$$E_k \simeq 8 \times 2^8 (400 + 250 + 200 + 200) \simeq 2,2 \times 10^6. \quad (14)$$

6.5 Some ideas for extra protections

The scheme for this attack does not depend on the configuration for the secret point, whether it is P or Q. The only aspect which comes into play is the choice of the pairing, and more exactly from the choice of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

6.5.1 Order of points P and Q

With a thoughtful choice of the parameters order, the secret integer can be 12 times more difficult to recover. More precisely, a point P in $\mathbb{G}_1 = E(\mathbb{F}_p)[r]$ has three coordinates (in Jacobian representation) which are simply integers on $|p|_2$ bits, while the point Q, in $\mathbb{G}_2 \subset E\left(\mathbb{F}_{p^{12}}\right)[r]$, has three coordinates which are elements of $\mathbb{F}_{p^{12}}$ (or $\mathbb{F}_{p^2}$ in the case of the use of a twisted curve). Thus, if $Q \in \mathbb{G}_2$ is secret, and if we target at first $x_Q \in \mathbb{F}_{p^{12}}$, then it is necessary to perform the attack 12 times in order to find the 12 integers of $\mathbb{F}_p$ that describe x_Q.

Besides, the point Q is made of $12 \times |p|_2$ bits, while P is contains $2 \times |p|_2$ bits. As a consequence, there are 6 times more bits to be found if Q is the secret rather than P.

6.5.2 Points representations

So far, we have seen that to choose Q as the secret is better from the defender point of view. Let us now analyze the point representations.

The structure of the Miller algorithm is such that points P and Q have an asymmetric role. Instead of only randomizing the public point, as proposed in the literature for cost reasons, it becomes interesting to

study the scenario in which the randomization is used on both points. To counter the effect of the collision attack, both masks have to be different.

Algorithm 7: Randomisation of both point P and Q with two different masks

Input : An elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$, points $P \in E\left(\mathbb{F}_{p^k}\right)[r]$ and $Q \in E\left(\mathbb{F}_{p^k}\right)$.
Output : Randomised Jacobian coordinates of P and Q.

1 $\lambda_1 \in \mathbb{F}_p^\star$ is randomly chosen ;
2 $P = (X_T : Y_T : Z_T)$ with $X_P \leftarrow x_P\lambda_1^2; \quad Y_P \leftarrow y_P\lambda_1^3; \quad Z_P \leftarrow \lambda_1$;
3 $\lambda_2 \in \mathbb{F}_p^\star$ is randomly chosen ;
4 $Q = (X_Q : Y_Q : Z_Q)$ with $X_Q \leftarrow x_Q\lambda_2^2; \quad Y_Q \leftarrow y_Q\lambda_2^3; \quad Z_Q \leftarrow \lambda_2$;
5 **return** P *and* Q ;

Now that both points are randomized (Algorithm 7), the coordinates system is not mixed any more. In this case, the projective representation allows to perform the operations with a minimal cost. When the representation of the points changes, it is necessary to re-compute the cost of mandatory operations (point addition, point doubling, line and tangent evaluation). The various possible combinations are studied and compared in the thesis [20]. In the Miller algorithm, on one hand we have the tangent line computation and the point doubling, and on the other hand the line evaluation and the addition of two points on the elliptic curve. These pairs of operations are actually very similar and can be computed together to avoid redundant operations.

In this section we are interested in representations that can be randomized, that is, projective and Jacobian for the three points P, T and Q. According to the comparative study in the thesis [20], the least expensive is the full-projective representation. These operations are detailed in Equations 15 and 16 with denominator elimination.

The doubling point T combined with the tangent evaluation $l_{T,T}(Q)$ becomes:

$$\begin{aligned} D &= 3X_T^2 + aZ_T^2 \\ F &= Y_T Z_T \\ G &= FY_T X_T \\ H &= D^2 - 8G \\ X_{[2]T} &= 2FH \\ Y_{[2]T} &= D(4G - H) - 8(Y_T F)^2 \\ Z_{[2]T} &= 8F^3 \\ l_{T,T}(Q) &= 2F(Z_T Y_Q - Y_T Z_Q) - D(X_Q Z_T - X_T Z_Q). \end{aligned} \tag{15}$$

The combined computation of $T + P$ and the line evaluation $l_{T,P}(Q)$ become:

$$\begin{aligned} A &= Y_T Z_P - Y_P Z_T \\ B &= X_T Z_P - X_P Z_T \\ C &= A^2 Z_T Z_P - B^3 - 2B^2 X_P Z_T \\ X_{T+P} &= BC \\ Y_{T+P} &= A(B^2 X_T Z_P - C) - Y_T Z_P B^3 \\ Z_{T+P} &= B^3 Z_T Z_P \\ l_{T,P}(Q) &= B(Z_T Y_Q - Z_Q Y_T) - A(Z_T X_Q - Z_Q X_T). \end{aligned} \tag{16}$$

The threat is now transferred to the randomization operation itself, it must be implemented in a secure way. The distribution of the mask is assumed to be uniform, that is $\lambda \in \mathbb{F}_p^\star$ is randomly chosen, which spells $\mathbb{P}(\Lambda = \lambda) = \frac{1}{p-1}$, where Λ denotes the discrete random variable.

We have already seen that $x_Q\lambda$ takes a random (unpredictable) value in a set of $p - 1$ elements. In other words, the secret is not manipulated any more. The only remaining target is actually the randomization step.

7 Conclusion and perspectives

In this paper, we recall some basic definitions about pairing-based cryptography in order to illustrate the fact that identity-based cryptography can be vulnerable to Side-channel attacks.

We first present pairings and their application to innovative cryptographic protocols, such as Identity-based Encryption. We provide the equations for lines and tangents to efficiently compute pairings by using Jacobian coordinates. The Identity-based Encryption by Boneh-Franklin [7] is our case study to present some side-channel attacks against pairings implementation. Our attacks can be successful for any pairing-based protocol involving a secret as input of the pairing.

Among the state of the art of the side-channel attacks, there are only a small number of practical validations. We describe a classical attack scheme against the pairing and perform the attack on an ARM Cortex-M3. The modular multiplication is the target because it involves an operation between known and unknown integers. Our work shows a slight difference between the attack paths, depending on the fact that the secret is the first or the second operand of the pairing. Besides, we are interested in the optimization of such attack. We show the characterization of the multiplication between two machine words. Some practical results show the efficiency of such an attack path. We also study the feasibility of a more advanced attack than a CPA. We drastically reduce the number of necessary traces and

also decrease the memory required for the attack. We perform the first horizontal CPA attack against an Ate pairing. We open the discussion on the consequences of our attack for state of the art secure implementations of PBC. According to [2], for the 128 bits security level, the KSS16 curves should provide the most efficient pairing implementations. Our horizontal attacks would then require 2 traces with 2 different inputs and the same secret key in order to be successful. We leave as future works the extension of our attacks for dedicated modules in embedded devices designed for higher sizes: 128, 256, 512, 1024 and 2048 bits architectures.

To circumvent classical attacks, the literature has proposed some countermeasures. We show that a practical collision attack defeats the less expensive protections. The practical results of a chosen plaintext collision attack allow us to conclude that the attack of the protected implementations induces a factor 10000 for the number of traces.

Here are some alternative solutions for a secure pairing implementation:

- Arithmetic randomization. For example in Residue Number System (RNS) representation with a new base which is calculated for every execution of the pairing. Of course, this randomization step will involve extra computation, as a future work we will study the risk-benefit ratio of this countermeasure.
- The randomization step which we wish to protect is made of three modular multiplications $x_Q\lambda, y_Q\lambda$ and $z_Q\lambda$. The protection of the modular multiplication was proposed in [4,9,30]. These protections can be used in order to secure the three critical multiplications which manipulate the secret coordinates.
- Consider slower external countermeasures to pairings such as $e(P,Q) = e(aP,bQ)$ or $e(P,Q) = e(P+R,Q)e(-R,Q)$, as proposed by Page and Vercauteren [32].

References

1. R. Azarderakhsh, D. Fishbein, G. Grewal, S. Hu, D. Jao, P. Longa, and R. Verma. Fast software implementations of bilinear pairings. *IEEE Trans. Dependable Sec. Comput.*, 14(6):605–619, 2017.
2. R. Barbulescu and S. Duquesne. Updating key size estimations for pairings. *Journal of Cryptology*, Jan 2018.
3. P. S. L. M. Barreto and M. Naehrig. Pairing-Friendly Elliptic Curves of Prime Order. SAC'05, pages 319–331, 2005.
4. A. Bauer, E. Jaulmes, E. Prouff, and J. Wild. Horizontal and Vertical Side-Channel Attacks against Secure RSA Implementations. In *Cryptographers' Track at the RSA Conference*, pages 1–17. Springer, 2013.
5. J.-L. Beuchat, J. E. González-Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez-Henríquez, and T. Teruya. High-speed software implementation of the optimal ate pairing over barreto–naehrig curves. In *ICPBC*, pages 21–39. Springer, 2010.
6. J. Blömer, P. Günther, and G. Liske. Improved Side Channel Attacks on Pairing Based Cryptography. *COSADE*, pages 154–168, 2013.
7. D. Boneh and M. Franklin. *Identity-Based Encryption from the Weil Pairing*, volume 32. Springer Berlin Heidelberg, 2001.
8. E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In *CHES*, pages 16–29. Springer, 2004.
9. C. Clavier, B. Feix, G. Gagnerot, C. Giraud, M. Rousselet, and V. Verneuil. ROSETTA for Single Trace Analysis. In *International Conference on Cryptology in India*, pages 140–155. Springer, 2012.
10. J. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. *CHES*, pages 292–302, 1999.
11. J.-S. Coron, P. Kocher, and D. Naccache. Statistics and secret leakage. In *Financial Cryptography*, pages 157–173. Springer, 2000.
12. Y. Desmedt and M. Burmester. Identity-based key infrastructures (iki). In *SEC*, pages 167–176. Springer, 2004.
13. J.-G. Dumas, P. Lafourcade, and P. Redon. *Architectures PKI et communications sécurisées*. Dunod, 2015.
14. R. Dutta, R. Barua, and P. Sarkar. Pairing-based cryptographic protocols: A survey. *IACR Cryptology ePrint Archive*, 2004:64, 2004.
15. I. Duursma and H. Lee. Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$. *Advances in cryptology - AsiaCrypt 2003*, 4:111–123, 2003.
16. N. El Mrabet, G. Di Natale, Flottes, and M. Lise. A Practical Differential Power Analysis Attack Against the Miller Algorithm. *PRIME*, pages 308–311, 2009.
17. S. Ghosh and D. Roychowdhury. Security of prime field pairing cryptoprocessor against differential power attack. pages 16–29. Springer, 2011.
18. B. Gierlichs, K. Lemke-Rust, and C. Paar. Templates vs. Stochastic Methods. In *CHES*, pages 15–29. Springer, 2006.
19. F. Hess, N. P. Smart, and F. Vercauteren. The Eta pairing revisited. *IEEE Transactions on Information Theory*, 52:4595–4602, 2006.
20. D. Jauvart. *Sécurisation des algorithmes de couplages contre les attaques physiques*. PhD thesis, Université Paris-Saclay, 2017.
21. D. Jauvart, J. J. Fournier, N. El Mrabet, and L. Goubin. Improving Side-Channel Attacks against Pairing-Based Cryptography. *CRiSIS*, 2016.
22. D. Jauvart, J. J. Fournier, and L. Goubin. First Practical Side-Channel Attack to Defeat Point Randomization in Secure Implementations of Pairing-Based Cryptography. In *ICETE, SECRYPT*. INSTICC, ScitePress, 2017.
23. A. Joux, A. Odlyzko, and C. Pierrot. The Past , evolving Present and Future of Discrete Logarithm. pages 1–23, 2014.
24. T. Kim and R. Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In *CRYPTO (1)*, volume 9814 of *Lecture Notes in Computer Science*, pages 543–571. Springer, 2016.
25. T. H. Kim, T. Takagi, D.-G. Han, H. W. Kim, and J. Lim. Side Channel Attacks and Countermeasures on Pairing Based Cryptosystems over Binary Fields. *Cryptology and Network Security*, pages 168–181, 2006.
26. C. K. Koc, T. Acar, and B. S. Kaliski. Analyzing and Comparing Montgomery Multiplication Algorithms. *IEEE micro*, 16(3):26–33, 1996.

27. P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. *Advances in Cryptology - CRYPTO'99*, pages 1–10, 1999.
28. T. Kusaka, S. Joichi, K. Ikuta, M. A.-A. Khandaker, Y. Nogami, S. Uehara, N. Yamai, and S. Duquesne. Solving 114-bit ecdlp for a barreto-naehrig curve. In H. Kim and D.-C. Kim, editors, *Information Security and Cryptology – ICISC 2017*, pages 231–244, Cham, 2018. Springer International Publishing.
29. R. Mayer-Sommer. Smartly analyzing the simplicity and the power of simple power analysis on smartcards. In *CHES*, pages 78–92. Springer, 2000.
30. M. Medwed and C. Herbst. Randomizing the Montgomery Multiplication to Repel Template Attacks on Multiplicative Masking. *COSADE, Lecture Notes in Computer Science*, 9, 2010.
31. V. S. Miller. Short Programs for functions on Curves. *Unpublished manuscript*, 97:101–102, 1986.
32. D. Page and F. Vercauteren. Fault and Side-Channel Attacks on Pairing Based Cryptography. 2004.
33. W. Pan and W. Marnane. A correlation power analysis attack against Tate pairing on FPGA. *Reconfigurable Computing: Architectures, Tools and Applications*, pages 340–349, 2011.
34. J.-J. Quisquater and D. Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *Smart Card Programming and Security*, pages 200–210. Springer, 2001.
35. M. Scott. Computing the Tate pairing. *CT-RSA*, pages 293–304, 2005.
36. M. Scott, N. Benger, M. Charlemagne, L. J. D. Perez, and E. J. Kachisa. On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves. In *International Conference on Pairing-Based Cryptography*, pages 78–88. Springer, 2009.
37. A. Shamir et al. Identity-based cryptosystems and signature schemes. In *Crypto*, volume 84, pages 47–53. Springer, 1984.
38. T. Unterluggauer and E. Wenger. Practical Attack on Bilinear Pairings to Disclose the Secrets of Embedded Devices. *ARES*, pages 69–77, 2014.
39. C. Whelan and M. Scott. Side Channel Analysis of Practical Pairing Implementations: Which Path Is More Secure? *VIETCRYPT 2006*, pages 99–114, 2006.