12 Physical Attacks

Nadia El Mrabet
EMSE

Louis Goubin
UVSQ

Sylvain Guilley
Telecom ParisTech & Secure-IC

Jacques Fournier
CEA Tech

Damien Jauvart
UVSQ/CEA Tech

Martin Moreau
Telecom ParisTech & Secure-IC

Pablo Rauzy
Université Paris 8

Franck Rondepierre
Oberthur Technologies

The security of modern cryptography is based on the impossibility of breaking the implemented algorithms in practice. In order to reach such a goal, the algorithms are built in such a way that breaking them in theory is as expensive as doing an exhaustive search on the whole key. The cryptosystems are made public to ensure a high knowledge of potential attacks and the key length is selected according to the existing computation power available to prevent any brute force of the key. The strength of a given algorithm grows exponentially with the length of the key used.

However, if by any means someone can access some part of the key and test whether he guessed the correct value of the key, independently from the rest of the key, he would be able to brute force the key. Indeed, if the size of such parts is small enough, the exhaustive search for each part will be practically feasible and by repeating the attack on the different parts, the cost of finding the whole key will grow linearly with the length of the key.

The study of whether it is possible to access small parts of the key or not has been a new field in cryptographic engineering since the middle of the nineties. This has been made possible thanks to a class of attacks called *Physical Attacks* against the implementations of cryptographic algorithms.

Physical attacks exploit the underlying intrinsic weaknesses of the integrated circuits used to run the cryptographic algorithms. In the context of cryptographic engineering, two types of physical attacks are of special interest. The first one, called *Side-Channel Analysis*, is based on the non-invasive measurement of side-channel information (power, electromagnetic, timing, temperature...) leaked during cryptographic computations. The second one, called *Fault Attacks*, consists of semi-invasively stressing the integrated circuit running the cryptographic algorithm (using lasers, clock or power glitches, or electromagnetic pulses, for example) to corrupt the calculations.

In this chapter, we shall see to what extent physical attacks have been successful so far in attacking implementations of pairing calculations. Both side-channel and fault attacks are covered. We also look at the countermeasures that have to be added to pairing calculations to increase their robustness against such attacks.

12.1 Side-Channel Attacks

The integrated circuits running the cryptographic algorithms are mostly made of transistors whose switching is directly correlated to the data being manipulated by the circuit. The difference in the switching activities of transistors when manipulating, say, a '0' or a '1' gives rise to measurable physical characteristics that provide the so-called side-channel information leakage. Those measurable physical characteristics can be, for example, timing information, power consumption, or electromagnetic emissions.

In order to capture the power consumption during the execution of algorithms, a small resistor is placed in series with the power ground input. The measured power traces can have a shape, like the time of duration, that depends on the program's inputs (Figure 12.1).

Kocher, Jaffe, and Jun introduced the power analysis as a means of side-channel attacks against cryptographic algorithms [39]. The main assumption of a power analysis attack is based on the fact that the power traces are correlated to the instructions performed by the device. Thus, studying the power traces can allow us to recover information about the instructions and data registers, and then about the involved operands.

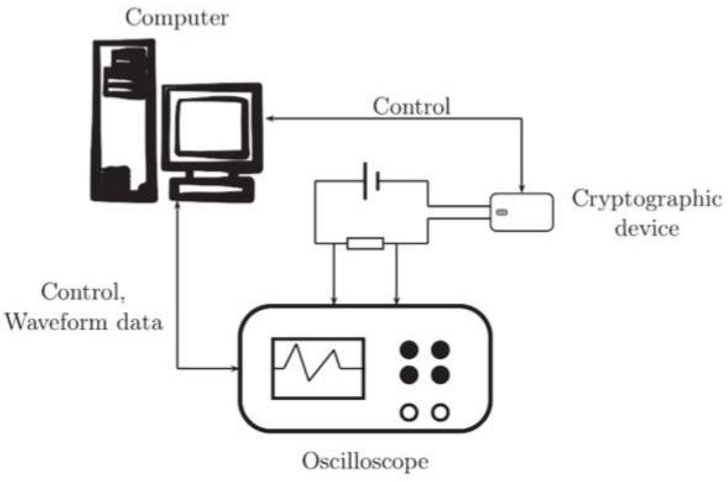

FIGURE 12.1 Setup for power analysis.

Physical side-channel leakage is not restricted to power consumption. The electric current also generates an electromagnetic (EM) field that can be captured with a small probe placed close to the part of the targeted circuit. Such a technique has the advantage of allowing an access restricted to some module (AES or big integer coprocessor, for instance) that limits the noise induced by uncorrelated operations.

In order to mount such attacks, the device shall be mounted on a dedicated board that has to be adapted to the form factor of the original target: bank chip card, sim card, secure element of a smartphone, rfid tag, ... Besides, especially for electromagnetic analysis, some chip preparation may be required, such as removing the black epoxy and the glue.

12.1.1 Simple Side-Channel Analysis

Timing attacks

Timing attacks were introduced by P. Kocher in 1996 and are the first known example of side-channel analysis. They are based on the fact that — in a given implementation — some predictable variation of the computational time may depend on the inputs (and in particular the secret key).

All 'basic' or 'straight-forward' implementations of cryptosystems can potentially succumb to such an attack. For instance, for a scalar ECC multiplication using the "double-and-add" method, the "add" operation, implemented with the Montgomery technique, may have different durations (according to the presence — or not — of a final subtraction). Therefore, by partitioning the inputs into two sets, the adversary can learn — thanks to the timing attack — whether this "add" operation is executed or not, and thus deduce the value of a secret key bit.

Note that symmetric algorithms can also be threatened by timing attacks, as illustrated in [10]. This type of attack is considered to be very powerful, mainly due to its low cost.

In the case of pairing-based cryptography, the Miller loop usually does not contain any conditional operations that depend on the secret data. However, at a lower level, the implementation may involve a basic operation (for instance, multiplication on a finite field) whose computational time depends on a secret bit, as highlighted, for instance in [37]. Therefore, timing attacks have also to be taken into account in the context of pairing-based cryptography. However, the required countermeasures appear to be the same as those developed to resist DPA-like attacks; we therefore refer to the sections about DPA.

Simple power analysis

Following Kerckhoff's principles, one may assume that the implemented cryptographic algorithms are publicly known. This is a legitimate assumption if we take into account the following facts: the publication of new algorithms within the cryptography community and the possibility of analyzing side-channel emissions coming from the algorithms' executions. Indeed, by looking at the power or execution trace of an algorithm, one can quickly recognize some patterns, and figure out which operations are being executed; it is especially true in public key cryptography where expensive modular operations are usually needed.

But the study of traces of execution is more powerful than simply giving access to the implemented algorithms. Such analyses, referred to as Simple Power Analysis (SPA) (since the first acquisitions were power consumption traces), are also a threat against weak implementations whose traces are dependant on the secret value. And this is an actual threat, since straightforward and fastest implementations won't thwart this dependency. A classic example is the study of a scalar multiplication $[n]P$ based on the double-and-add algorithm; see Algorithm 12.1. In this algorithm, we can see that an addition $Q+P$ is only performed if the secret bit n_i is equal to 1. So, if the trace of a doubling is different from the trace of the addition — and this is the case for the Short-Weierstrass formulae — the attacker will have direct access to the secret scalar n, which otherwise would have required him or her to solve the discrete logarithm problem.

ALGORITHM 12.1 Double-and-add algorithm.

```
Input  : $n = (n_{t-1}, \dots, n_0)_2$ and $P \in E$
Output : $[n]P$
$Q \leftarrow P$
for $i = t-1$ downto 0 do
    $Q \leftarrow [2]Q$                                  // double
    if $n_i == 1$ then
        $Q \leftarrow Q + P$                             // add
    end
end
return $R_0$
```

Several countermeasures have been proposed in the literature to thwart such attacks. For the case of scalar multiplications, which is pretty similar to the exponentiation case, the flaw is twofold: the branch conditionally selected from a secret and also a different implementation of the doubling and addition operations. This last point also applies to exponentiations that use optimized modular squaring. Existing countermeasures consist of either solving the flaw or in changing the algorithm in order to perform a regular flow of operations, independent from the secret. For the first solution we can cite the atomicity principle [14] and for the second one we can cite the Montgomery ladder [35] (see Algorithm 12.2). Other solutions exist, but have larger impacts on the performances (timing) of implemented algorithms.

ALGORITHM 12.2 Montgomery ladder.

```
Input  : $n = (n_{t-1}, \dots, n_0)_2$ and $P \in E$
Output : $[n]P$
$R_0 \leftarrow P$
$R_1 \leftarrow [2]P$
for $i = t-1$ downto 0 do
    $b \leftarrow n_i$
    $R_{1-b} \leftarrow R_0 + R_1$
    $R_b \leftarrow [2]R_b$
end
return $R_0$
```

Besides, the whole security does not rely only on software countermeasures. At the hardware level, techniques such as clock jitters, additional power noise, dummy cycles, or power filtering help to increase the resistance against (simple) side-channel analysis.

12.1.2 Advanced Side-Channel Analysis

Differential power analysis

Differential Power Analysis (DPA) was initially defined by Kocher, Jaffe, and Jun [39] to target the Data Encryption Standard (DES). In the family of differential analysis attacks, we include, for example, the differential power and electromagnetic attacks. Differential power analysis works on several power/EM traces that are analyzed using statistical tools, which helps in getting rid of variations due to data manipulated, and some noise, which are embarrassing problems in the case of a single trace.

The principle is to build, for the system under attack, a 'function' parameterized by a small part of the algorithm that we want to attack. The aim is to recover the set 'function' corresponding to the secret. For this we acquire a large number of images of the 'function.' Furthermore, we construct a theoretical series of corresponding images for each set function. Then we choose a distinguisher to compare theoretical series and series from the acquisition. There are many such distinguishers, the main ones being difference of means, correlation coefficient, and mutual information.

In the case of public key cryptography, most classical DPA attacks target a scalar multiplication operation with the aim of recovering the scalar bits one by one. The description of a DPA attack against ECC is well introduced in [32].

Assume that the double-and-add method is implemented with one of the regular variants given in Algorithm 12.1. Let $n = (n_{t-1}, \ldots, n_0)_2$ be the scalar multiplier. Suppose that an attacker already knows the most significant bits, $n_{t-1}, \ldots, n_{j+1}$. Then, the attacker has to make a guess on the next bit n_j, which is equal to 1. He randomly chooses several points $P_1, \ldots, P_r$ and computes $Q_s = \left[\sum_{i=j}^{t-1} n_i 2^i\right] P_s$ for $1 \le s \le r$.

Using a boolean selection function g, the attacker prepares two sets: the first set, S_{true}, contains the points P_s such that $g(Q_s) = \texttt{true}$ and the second set, S_{false}, contains those such that $g(Q_s) = \texttt{false}$. Then, a candidate for the selection function may, for example, be the value of a given bit in the representation of Q_s.

Let $C^{(s)}$ denote the side-channel information associated to the computation of $[n]P_s$ by the cryptographic device (e.g., the power consumption). If the guess $n_j == 1$ is incorrect then the difference obtained in Equation 12.1 will be $\simeq 0$.

$$\left\langle C^{(s)} \right\rangle_{\substack{1 \le s \le r \\ P_s \in S_{\text{true}}}} - \left\langle C^{(s)} \right\rangle_{\substack{1 \le s \le r \\ P_s \in S_{\text{false}}}} . \tag{12.1}$$

If the guess is wrong, both sets appear as two random sets, otherwise the guess is correct. After revealing n_j, the remaining bits $n_{j-1}, \ldots, n_0$ are recovered recursively by the same method.

Correlation power analysis

In DPA, the classification of power traces is based on comparing the differences between the measured traces. Brier, Clavier, and Olivier in 2004 at CHES proposed an improvement of DPA based on the use of Pearson's correlation for comparing the measured side-channel traces and a leakage model based on the Hamming Weight (HW) of the manipulated data.

The side-channel information of the device is supposed to be linear in $H(D \oplus R)$, the Hamming distance of the data manipulated D, with respect to a *reference state* R. The linear correlation factor is used to correlate the side-channel curves with this value $H(D \oplus R)$. The maximum correlation factor is obtained for the right guess of secret key bits.

Let C be the side channel (power consumption for instance) of the chip; its consumption model is:

$$W = \mu H(D \oplus R) + \nu . \tag{12.2}$$

The correlation factor $\rho_{C,H}$ between the set of power curves C and values $H(D \oplus R)$ is defined as: $\rho_{C,H} = \frac{cov(C,H)}{\sigma_C \sigma_H}$.

The principle of the attack is then the following:

- Perform r executions on the chip with input data $m_1, \ldots, m_r$ and collect the corresponding power curves $C^{(1)}, \ldots, C^{(r)}$.
- Predict some intermediate data D_i as a function of m_i and key hypothesis g.

- Produce the set of the r predicted Hamming distances: $\{H_{i,R} = H(D_i \oplus R), i = 1, \ldots, r\}$.
- Calculate the estimated correlation factor:

$$\widehat{\rho_{C,H}} = \frac{r \sum C^{(i)} H_{i,R} - \sum C^{(i)} \sum H_{i,R}}{\sqrt{r \sum (C^{(i)})^2 - (\sum C^{(i)})^2} \sqrt{r \sum H_{i,R}^2 - (\sum H_{i,R})^2}}. \tag{12.3}$$

When the attacker makes the right guesses for values of the reference state R and secret leading to data D, the correlation factor ρ is maximum.

This attack is more powerful than DPA in the sense that the 'leakage' peaks are generally more visible in CPA with the same conditions as in DPA.

12.1.3 Side-Channel Attacks against Pairings

In the case of pairings, side-channel attacks are relevant whenever pairings are used in schemes involving some secret data, which is typically the case when pairings are used in identity-based encryption schemes.

The fundamental idea of identity-based encryption is to allow the user's public key to be a public function of his identity. This requires a trusted authority (T_A) that sends him his private key. This trusted authority creates all the private keys related to an Identity-Based (IB) protocol. The advantage of IB is to simplify the transmission of public keys while sending the encryption of a message. Indeed, it is no longer necessary to use certificates or public-key infrastructure (PKI), since the public key used for encryption can be deterministically (and publicly) deduced from the identity of the receiver.

The important point during an IB protocol is that the decryption involves a pairing computation between the private key of the user and a public key. We call the public key the part of the message used during the pairing calculation involving the secret key. A potential attacker can know the algorithm used, the number of iterations, and the exponent. The secret is only one of the arguments of the pairing. The secret key influences neither the time execution nor the number of iterations of the algorithm, which is different from RSA or ECC protocols.

From here on, the secret will be denoted P and the public parameter (or the point used by the attacker) Q. We are going to describe a DPA attack against the Miller algorithm. We restrict this study to the case where the secret is used as the first argument of the pairing. If the secret is used as the second argument, the same attack can be applied; this assumption is shown theoretically and practically in [21] and also in [56]. We assume that the algorithm is implemented on an electronic device such as a smart card and used in a protocol involving IB cryptography. The attacker can send as many known entries Q for the decryption operation of IBC as he wants, and he can collect the power consumption curves.

Most pairing computations are based on the use of the Miller algorithm. This is in particular true for the Weil, Tate, and Ate pairings. We assume that the Miller algorithm is implemented in software running on an electronic device: for example, a smart card. The attacks are performed during the execution of a cryptographic protocol based on identity. Let Q be the public message. The private key will be represented by the point P in the computation of the pairing $e(P, Q)$. We restrict the study to the case where the secret is the first argument of the pairing. Placing the first secret of the coupling parameter is a first countermeasure against some side-channel attacks as proposed in [60]. If the secret is the second argument of the pairing, the same attack patterns may apply and allow us to recover the secret used. The attacker can compute as many times as necessary pairings between the secret P (that will not change) and inputs Q (that changes at will). He can record and store the power consumption curves for each of those computations, together with the final result of the Miller algorithm.

Description of the attack

When implementing pairings, different coordinate systems may be used. This does not have any significant impact on the feasibility of side-channel attacks. Indeed, in the Miller loop, even if the choice of the coordinate system will give rise to different implementations of the 'lines' and 'tangents' computations, the underlying internal operations will be the same modular multiplications and additions on long precision numbers.

As described in the general DPA/CPA approach, we try to identify some operations that involve a secret and a known operand; such operations are in bold in the following equations. As already explained, there are several ways of implementing the Miller loop. For example, [60] takes the case of affine coordinates; in this case the line and the tangent equations are the following.

The line equation is the formula to compute $l_{T,P}(Q)$, the line passing through T and P evaluated in Q is:

$$l_{T,P}(Q) = \boldsymbol{y_Q - y_T} - \frac{y_P - y_T}{x_P - x_T}(\boldsymbol{x_Q - x_T}).$$

The tangent equation is $l_{T,T}(Q)$, the tangent line through point T evaluated in Q. This equation is:

$$l_{T,T}(Q) = \boldsymbol{y_Q - y_T} - \frac{3x_T^2 + a}{2y_T}(\boldsymbol{x_Q - x_T}).$$

The case for Jacobian coordinates is treated by [21] and [23] with the same aim of targeting an operation in order to recover one coordinate of the secret input point. If the points are a three-tuple, then it is necessary to recover a second component. Now we use the elliptic curve equation to find the last coordinate. The secret point is recovered.

The line and tangent equation in Jacobian coordinates are the following:

$$l_{T,P}(Q) = \frac{2\boldsymbol{y_Q y_T} z_T^4 - 2y_T^2 \boldsymbol{z_T z_Q^3} + (y_P z_T^3 - y_T z_P^3)(x_T \boldsymbol{z_T z_Q^3} - x_Q \boldsymbol{z_Q z_T^3})}{2y_T \boldsymbol{z_T^4 z_Q^3}}$$

and

$$l_{T,T}(Q) = \frac{2y_T(\boldsymbol{y_Q z_T^3} - \boldsymbol{y_T z_Q^3}) - z_Q(3x_T^2 + az_T^4)(\boldsymbol{x_Q z_T^2} - \boldsymbol{x_T z_Q^2})}{2y_T \boldsymbol{z_T^3 z_Q^3}}.$$

The same approach works when in mixed coordinates as described in [21] and [11]. For optimization reasons it is also possible to mix system coordinates. The equations are available in [21]. Let $T = (X_T, Y_T, Z_T)$ be a point in Jacobian coordinates, P and Q in affine coordinates, then the line and tangent equation in mixed coordinates are the following:

$$l_{T,T}(Q) = 2\boldsymbol{y_Q Y_T} Z_T^3 - 2Y_T^2 - (3X_T^2 + aZ_T^4)(\boldsymbol{x_Q Z_T^2} - X_T)$$

and

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

Multiplication in $\mathbb{F}_q$

We describe the attacks as if we have the embedded degree $k = 1$, and then the coordinates of Q being elements of $\mathbb{F}_q$. This way, the targeted multiplication $Z_P^2 x_Q$ is a multiplication in $\mathbb{F}_q$. The DPA attack also works when $k > 1$. Even if the multiplication $Z_P^2 x_Q$ becomes a multiplication between an element of $\mathbb{F}_q$ and an element of $\mathbb{F}_{q^k}$, we can consider a multiplication between two $\mathbb{F}_q$ elements.

Indeed, $x_Q \in \mathbb{F}_{q^k}$ is written: $x_Q = \sum_{i=0}^{k-1} x_{Q_i} \xi^i$, with $(1, \xi, \xi^2, \ldots, \xi^{k-1})$ a basis of $\mathbb{F}_{q^k}$, and there exists a polynomial R such that $deg(R) = k$ with ξ root of R, $(R(\xi) = 0)$. Then

$Z_P^2 x_Q = \sum_{i=0}^{k-1} \left(Z_P^2 \times x_{Q_i}\right) \xi^i$, is composed of k products in $\mathbb{F}_q$. So we can focus on one of these k products in $\mathbb{F}_q$ to apply the DPA attack as described.

In the same way, to compute the difference $(Z_P^2 x_Q - X)$, we compute a difference between elements of $\mathbb{F}_q$ as in the affine case.

Indeed, if $Z_P^2 x_Q = \sum_{i=0}^{k-1} (Z_P^2 x_Q)_i \xi^i$ then

$$Z_P^2 x_Q - X = \left((Z_P^2 x_Q)_0 - X\right) + \sum_{i=1}^{k-1} (Z_P^2 x_Q)_i \xi^i.$$

Targeting the first iteration in the Miller loop

We describe the attack for the first iteration. It is the simplest case, because we know that for this iteration, $T = P$. We can provide the attack for the j^{th} iteration. For this iteration we find $T = [j]P$, where $[j]P$ represents the scalar multiplication of point P by the integer j.

We know l, the order of the point Q (as P and Q have the same order). By counting the number of clock cycles, we can find the number d of iterations we have made before the DPA attack. Then, reading the binary decomposition of l directly gives us j. We consider that at the beginning $j = 1$, if $l_{n-1} = 0$ then $j \leftarrow 2j$, else $j \leftarrow 2j + 1$, and we go on, until we arrive at the $(n - 1 - d)^{th}$ bit of l.

If the attack is done during the j^{th} iteration of the Miller algorithm, we find the coordinates of $[j]P$. In order to find P, we just have to compute $j^{'}$, the inverse of j modulo l, and then $P = [j'][j]P$.

Furthermore, we present the attack against the basic Miller algorithm. The attack can be straightforwardly generalized to the optimised Miller algorithm given in [38].

Description of the attack

In order to retrieve the secret key $P = (X_P, Y_P, Z_P)$, the circuit has to be used to perform some calculations while the power consumption of the physical device is monitored. In particular, the measurement of the consumed power must be done during a time slot when the circuit calculates a result that depends on both the secret key and some controllable input data.

For example, we decided to observe the power consumption when the circuit performs the multiplication between Z_P^2 (a part of the secret key) and x_Q (the input data). This operation is done during the second control step. To retrieve the second part of the key (X_P), we focused on the subtraction between the previously performed multiplication and the key.

The DPA attack against the Miller algorithm was first proposed by Page and Vercauteren [44]. Over the years, the proposed schemes have been enhanced. Reference [60] extends the attack to several other operations and proposes a scheme using CPA. Another remarkable improvement is proposed by [11], where the authors attack the modular addition and multiplication of elements in a finite field of large prime characteristics. In this chapter, we present those attacks against pairings and provide simulation results.

To implement the attack, it is necessary to target an operation in the line or tangent equation. Let $\star$ be the general targeted operator between $g \in \mathbb{F}_q$ and $U \in E(\mathbb{F}_q)$. For instance, $g \star U$ can be $g - U_x \in \mathbb{F}_q$.

The attack scheme proposed against Miller is Algorithm 12.3.

The last part of the key (Y_P) can be mathematically inferred from X_P and Z_P^2. Indeed, the elliptic curve equation $E : Y^2 = X^3 + aXZ^4 + bZ^6$ is a quadratic equation in Y_P. The square root of $\sqrt{X_P^3 + aX_P Z_P^4 + bZ_P^6}$ gives us two possibilities for the value of Y_P; testing them by an execution of the Miller algorithm will give the correct coordinates for P.

ALGORITHM 12.3 A Messerges-style DPA attack to reveal $P = (x_P, y_P)$ by guessing y_P **one bit at a time.**

```
Input  : $n$ is the bitlength max of $y_P$
Output : A candidate for the coordinate $y_P$

Set $g$ to 0
for $i = 0$ upto $n - 1$ do
    Set $S_{hi}$ and $S_{lo}$ to empty
    Guess the $i^{\text{th}}$ bit of $g$ to one
    for $k = 0$ upto $r - 1$ do
        Select at random a point $U$ of $E$
        Calculate $X = g \star U$
        Use device to execute $e(P, U)$, collect power signal $S_k$
        if the $i^{th}$ bit of $X$ is 1 then
            add $S_k$ to $S_{hi}$
        else
            add $S_k$ to $S_{lo}$
        end
    end
    Average power signals to get DPA bias $D = \overline{S_{hi}} - \overline{S_{lo}}$
    if DPA bias signal has a spike then
        The guess was right: set $i^{\text{th}}$ bit of $g$ to 1
    else
        The guess was wrong: set $i^{\text{th}}$ bit of $g$ to 0
    end
end
return $g$
```

The practical feasibility of such attacks is illustrated in [56]. The target is an Ate pairing over BN curves $e(P, Q)$ with P the secret input. The targeted operation is a modular multiplication. To implement this over long integer ($\simeq 256$ bits), they use the Montgomery method. The device architecture imposes on the attacker to target 16 bits at time.

12.2 Fault Attacks

In 1984, A. Shamir challenged the cryptography community to find a protocol based on the user's identity [51]. This challenge was solved nearly twenty years later by D. Boneh and M. Franklin. In 2003, D. Boneh and M. Franklin created an identity-based encryption (IBE) scheme based on pairings [13]. The general scheme of an identity-based encryption is described in [13], and several protocols based on pairings have been developed since [33]. A feature of identity-based protocols is that a computation of a pairing involving the private key and the plaintext is performed in order to decipher a message. A pairing is a bilinear map e taking as inputs two points P and Q of an elliptic curve. The pairing computation gives the result $e(P, Q)$. Several pairings have been described in the literature. The Weil and the Tate pairing was developed [54] without any consideration for the efficiency of the computation. Once pairings were used to construct protocols, cryptographers sought more efficient algorithms. In chronological order, the Duursma and Lee algorithm [18], the Eta [8], Ate, twisted Ate [29], optimal pairings [57], and pairing lattices [28] were invented. Recently, a construction of pairing over a general abelian variety was proposed in [42]. The latest implementations results [3, 26, 49] of pairing computations are fast enough to consider the use of pairing-based protocols in embedded devices. Consequently, it seems fair to wonder if pairing-based protocols involving a secret are secure against physical attacks in general, and fault attacks in particular. We focus here on fault attacks against pairing-based cryptography.

Since 2006, several fault attacks against pairings have been proposed. Here we will present what are in our opinion the most significant ones. For each attack, we assume that the pairing is used during an identity-based protocol. The secret point is stored into an embedded electronic device that can be attacked with fault attacks. The location of the secret is not important in practice. Indeed, the equations that leak information about the secret can provide information as to whether the secret is the first or the second parameter. Often, the attack is easier when the secret is the second parameter. That is why we consider the cases where the first parameter is the secret argument.

The necessary background in order to understand pairings and IBE is presented in Chapter 1. The first fault attack against a pairing was proposed by Page and Vercauteren [45] and is presented in Section 12.2.2. Then, we describe the adaptations of the previous attack against the Miller algorithm in Section 12.2.2. Whelan and Scott [59] highlighted the fact that pairings without a final exponentiation are more sensitive to a sign-change fault attack. After that, El Mrabet [19] generalized the attack of Page and Vercauteren to the Miller algorithm used to compute all the recent optimizations of pairings. Another method is adopted in [4], based on instruction skips, and presented in Section 12.2.2. In [40], Lashermes et al. proposed a fault attack against the final exponentiation during a Tate-like pairing. Their attack is described in Section 12.2.3. Finally, we conclude the description of fault attack in Section 12.2.4.

12.2.1 What Are Fault Attacks?

The goal of a fault attack is to inject errors during the calculation of an algorithm in order to reveal sensitive data. At first these attacks required a very precise positioning and expensive equipment to be performed, but now even some cheap equipment allows us to perform them [27]. The faults can be performed using a laser, an electromagnetic pulse, and power or clock glitches [16, 17, 36].

The effect of a fault can be permanent, i.e., a modification of a value in memory, or transient, i.e., a modification of data that is not stored into memory at one precise moment.

At the bit level, a fault can be a bit-flip if the value of a bit is complemented. Or it can be stuck-at (0 or 1) if the bit modification depends on its value.

The fault cannot only modify the data manipulated but also modify a program's execution. As an example in a microcontroller, if a fault occurs on the opcode and modifies it, the executed instruction will be modified. This method gives rise what is called an instruction skip fault model where an instruction is skipped by modifying its opcode to a value representing an instruction without effect (e.g., NOP).

12.2.2 Fault Attacks against the Miller Algorithm

In this section we present the existing attacks against the Miller algorithm. We describe in Section 12.2.2 an attack against the Duursma and Lee algorithm, since it was the first attack against a pairing and, more importantly, all the following attacks are constructed on this scheme. Then, in Section 12.2.2 , we describe the attacks against the Miller algorithm.

Attacks against the Dursma and Lee algorithm

The Duursma and Lee algorithm is not constructed using the Miller algorithm. But it was the first implementation of a pairing to be attacked. The attack was developed by Page and Vercauteren in [45].

Duursma and Lee [18] define a pairing over hyperelliptic curves, and in particular, over super-singular elliptic curves over finite fields of characteristic 3. For $\mathbb{F}_q$ with $q = 3^m$ and $k = 6$, suitable curves are defined by

$$E : y^2 = x^3 - x + b$$

with $b = \pm 1 \in \mathbb{F}_3$. Let $\mathbb{F}_{q^3} = \mathbb{F}_q[\rho]/(\rho^3 - \rho - b)$ and $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2 + 1)$. The distortion map $\phi : E(\mathbb{F}_q) \to E(\mathbb{F}_{q^6})$ is defined by $\phi(x, y) = (\rho - x, \sigma y)$. Then, with $\mathbb{G}_1 = \mathbb{G}_2 = E(\mathbb{F}_{3^m})$ and $\mathbb{G}_T = \mathbb{F}_{q^6}$, Algorithm 12.4 computes an admissible, symmetric pairing.

ALGORITHM 12.4 The Duursma-Lee pairing algorithm.

```
Input  : $P = (x_P, y_P) \in \mathbb{G}_1$ and $Q = (x_Q, y_Q) \in \mathbb{G}_2$.
Output: $e(P, Q) \in \mathbb{G}_3$.
$f \leftarrow 1$
for $i = 1$ upto $m$ do
    $x_P \leftarrow x_P^3,\ y_P \leftarrow y_P^3$
    $\mu \leftarrow x_P + x_Q + b$
    $\lambda \leftarrow -y_P y_Q \sigma - \mu^2$
    $g \leftarrow \lambda - \mu\rho - \rho^2$
    $f \leftarrow f \cdot g$
    $x_Q \leftarrow x_Q^{1/3},\ y_Q \leftarrow y_Q^{1/3}$
end
return $f^{q^3-1}$
```

The attack developed by Page and Vercauteren in [45] consists of modifying the number of iterations during the Duursma and Lee algorithm. The hypotheses to perform the attack are that

- the two inputs parameters (points P and Q) are fixed, one is secret and the other public;
- the pairing implementation is public;
- two pairing computations are done, one valid and one faulty.

The analysis of the quotient of the two results gives information about the secret. Indeed, the quotient of the two results cancel terms that are not influenced by the fault. Firstly, Page and Vercauteren described how to recover the secret point if the final exponentiation is not performed (i.e., Line 9 of Algorithm 12.4). Then they explained how to reverse the final exponentiation for a complete attack.

Attack without the final exponentiation

Let $P = (x_P, y_P)$ be the secret input during the pairing computation and let $Q = (x_Q, y_Q)$ be selected by the attacker. We consider the Duursma and Lee algorithm without the final exponentiation (Line 9).

Let $\overline{e}[\Delta]$ be the execution of Algorithm 12.4 where the fault replaces the loop bound m (in Line 2) with Δ. Then the result of the Duursma and Lee algorithm without the final exponentiation, instead of being a product of polynomials of the form

$$\prod_{i=1}^{m}\left[(-y_P^{3^i}\cdot y_2^{3^{m-i+1}}\sigma-(x_P^{3^i}+x_2^{3^{m-i+1}}+b)^2)-(x_P^{3^i}+x_2^{3^{m-i+1}}+b)\rho-\rho^2\right],$$

is a product of the form

$$\prod_{i=1}^{\Delta}\left[(-y_P^{3^i}\cdot y_2^{3^{m-i+1}}\sigma-(x_P^{3^i}+x_2^{3^{m-i+1}}+b)^2)-(x_P^{3^i}+x_2^{3^{m-i+1}}+b)\rho-\rho^2\right]$$

for a random integer Δ.

If $\Delta = m+1$, then recovering the secret point P is easy. We have two results

$$\begin{aligned} R_1 &= \overline{e}[m](P,Q) \\ R_2 &= \overline{e}[m+1](P,Q) \end{aligned}$$

where R_1 is correct and R_2 is faulty. Let $g_{(i)}$ be the i-th factor of a product produced by the algorithm. The quotient of the two results produces a single factor,

$$g_{(m+1)} = (-y_P^{3^{m+1}}\cdot y_2\sigma-(x_P^{3^{m+1}}+x_2+b)^2)-(x_P^{3^{m+1}}+x_2+b)\rho-\rho^2.$$

Given that $\forall z\in\mathbb{F}_q, z^{3^m}=z$, the attacker can easily extract x_P or y_P based on the knowledge of x_Q and y_Q.

In practice, the faulty result Δ cannot be forced to $m+1$. It is more realistic to assume that the fault gives $\Delta = m\pm\tau$ for a random unknown integer τ. As a consequence, the attacker computes two results

$$\begin{aligned} R_1 &= \overline{e}[m\pm\tau](P,Q) \\ R_2 &= \overline{e}[m\pm\tau+1](P,Q), \end{aligned}$$

and once again, considering the quotient, the attacker obtains a single term $g_{(m\pm\tau+1)}$.

In order to apply the same approach, the attacker should discover the exact value of τ. Indeed, this value is needed to correct the powers of x_P, y_P, x_Q, and y_Q. As the implementation of Duursma and Lee algorithm is supposed to be public, the number of operations performed during the faulty execution leaks the value of τ. Then the attack consists of several faulty executions of Algorithm 12.4, until we find two results R_1 and R_2 satisfying the requirements. The probability to obtain two values R_1 and R_2 after a realistic number of tests was computed in [19].

The probability to obtain two consecutive numbers after n picks among N integers is

$$P(n,N) = 1-\frac{B(n,N)}{C_{n+N}^{n}},$$

where

$$\left\{\begin{aligned} N\le 0, n>0, B(n,N) &= 0, \\ \forall N, n=0 B(n,N) &= 1 \\ B(n,N) &= \textstyle\sum_{j=1}^{N}\sum_{k=1}^{n} B(n-k,j-2). \end{aligned}\right.$$

For instance, for an 8-bit architecture only 15 tests are needed to obtain a probability larger than one half, $P(15,2^8)=0.56$, and only 28 for a probability larger than 0.9.

Reversing the final exponentiation

The attack described above is efficient without the final exponentiation. But since the final exponentiation is a part of the Duursma and Lee algorithm, Page and Vercauteren present a method to reverse it. The problem is that given the result $R = e(P, Q)$ the attacker wants to recover S, the value obtained in Line 7 of Algorithm 12.4, before the final exponentiation (i.e., $R = S^{q^3-1}$). Given R, the value of S is only determined up to a non-zero factor in $\mathbb{F}_{q^3}$. Indeed, the Fermat little theorem implies that $\forall c \in \mathbb{F}_{q^3} \setminus \{0\}$, $c^{q^3-1} = 1$. Furthermore, for one solution S of the equation $X^{q^3-1} - R = 0$, all the other solutions are of the form cS, for $c \in \mathbb{F}_{q^3} \setminus \{0\}$. At first sight, the attacker would not be able to choose the correct value S among the $q^3 - 1$ possibilities. However, given the description of the attack, the attacker does not need to reverse the powering of a full factor, but only a single factor with a special form:

$$R = \frac{R_2}{R_1} = \frac{\overline{e}[m \pm \tau + 1](P, Q)}{\overline{e}[m \pm \tau](P, Q)} = g_{(m \pm \tau+1)}^{q^3-1}.$$

We want to recover $g_{(m\pm\tau+1)}$, in order to find the coordinates of the secret point x_P and y_P: In order to solve this problem, Page and Vercauteren split it in two:

1. a method to compute one valid root of $R = g^{q^3-1}$ for some factor g, and
2. a method to derive the correct value of g from among all possible solutions.

The first problem is solved throughout the method of Lidl and Niederreiter [41] to compute roots of the linear operator $X^{q^3} - R \cdot X$ on the vector space $\mathbb{F}_{q^6}/\mathbb{F}_{q^3}$. They use a matrix representation of the problem to find all the solutions of the equation $X^{q^3-1} - R = 0$. Then, in order to find the correct root among the $q^3 - 1$ possibilities, Page and Vercauteren use the specific form of the factors in the product. Indeed, the terms $\rho\sigma$ and $\rho^2\sigma$ do not appear in the correct value and this gives a linear system of equations providing the solution. As the method to reverse the final exponentiation is specific to the Duursma and Lee algorithm, we do not give the equations. They are presented with examples in [22, 45].

Attacks against the Miller algorithm

A specific sign-change attack

The first attack against the Miller algorithm was developed by Whelan and Scott [59]. They use the same approach as the attack against Duursma and Lee. They compute two pairing values, one correct and one faulty. However, the fault is no longer on the Miller loop bound but into the Miller variable. Whelan and Scott analyze several pairings and study the success of the attack whether the secret is the point P or Q. They consider the case of the Eta pairing [8]. This pairing is defined over super-singular curves for small characteristics. Considering the recent result on the discrete logarithm problem [31] and the fact that the attack is based on the scheme of the Page and Vercauteren attack, we do not describe it. Whelan and Scott target the Weil pairing. First they try to describe a general fault model: any fault is injected during any iteration of the Miller algorithm. The attacker needs to solve a non-linear system and they conclude that it cannot be done. So they consider a more specific attack: a sign cannot change fault attack (a single sign bit is flipped [12]). They consider that the attacker modifies the sign of one of the coordinates of the point P or Q. This attack is the most efficient when exactly the last iteration of the Miller algorithm is corrupted. They consider the ratio between a valid and a faulty execution of the Weil pairing, and, using the equations, they obtain a linear system in the coordinates of the secret point. In this case, the attack is successful. If the fault is injected earlier in the Miller algorithm, the analysis is more complex, as several square and cubic roots have to be computed, but possible. Then they consider the Tate pairing. As the Tate pairing is also constructed using the Miller algorithm, the attack described for the Weil pairing should be efficient. However, due to the complex final exponentiation, they conclude that the Tate pairing is efficiently protected against the sign-change fault they propose.

A general fault attack

In [19], El Mrabet considers a fault attack based on the Page and Vercauteren attack [45]. The fault consists of modifying the number of iterations during the execution of the Miller algorithm. As the Miller algorithm is the central step for the Weil, the Tate, Ate, twisted Ate, optimal pairings, and pairing lattices, the fault model is valuable for a wide class of pairings. However, the attack targets only the Miller algorithm, the final exponentiation is not reversed cryptanalytically, and the author assumes that another attack could annihilate it. In Section 12.2.3 we describe a recent attack that reverses the final exponentiation. We describe here the general attack against the Miller algorithm. The difficulty of the attack relates to the resolution of a non-linear system.

El Mrabet considers that the number of iterations in the Miller algorithm is modified by a fault attack and denotes τ the new number of iterations. The value of τ is random but can be determined afterwards if the attacker knows the number of iterations, by monitoring the timing of the computation, for example. The aim is to obtain two consecutive results of Miller's algorithm $F_{\tau,P}(Q)$ and $F_{\tau+1,P}(Q)$. As in the attack on the Duursma and Lee algorithm, we consider the ratio $\frac{F_{\tau+1,P}(Q)}{F_{\tau,P}(Q)^2}$. Then an identification in the basis of $\mathbb{F}_{p^k}$ leads to a system that reveals the secret point.

Without loss of generality, we describe the attack when the embedding degree of the curve is $k = 4$. This allows the description of the equation. As the important point of the method is the identification of the decomposition in the basis of $\mathbb{F}_{p^k}$, it is easily applicable when k is larger than 3. Indeed, $k = 3$ is the minimal value of the embedding degree for which the system obtained can be solved. At the τ-th step, the Miller algorithm calculates $[j]P$. During the $(\tau+1)^{th}$ iteration, it calculates $[2j]P$, and considering the value of the $(\tau+1)^{th}$ bit of $\log_2(r)$, it either stops at this moment, or it calculates $[2j+1]P$.

Let $B = \{1, \xi, \sqrt{\nu}, \xi\sqrt{\nu}\}$ be the basis of $\mathbb{F}_{p^k}$; this basis is constructed using tower extensions. The point $P \in E(\mathbb{F}_p)$ is given in Jacobian coordinates, $P = (X_P, Y_P, Z_P)$, and the point $Q \in E(\mathbb{F}_{p^k})$ is in affine coordinates. As k is even, we can use a classical optimization in pairing-based cryptography, which consists of using the twisted elliptic curve to write $Q = (x, y\sqrt{\nu})$, with x, y and $\nu \in \mathbb{F}_{p^{k/2}}$ and $\sqrt{\nu} \in \mathbb{F}_{p^k}$ [6]. We will consider here only the case where $r_{\tau+1} = 0$. The case where $r_{\tau+1} = 1$ can be treated similarly is described in [19]. The non-linear system in the case $r_{\tau+1} = 1$ is a bit more complex and must be solved using the discriminant theory.

When $r_{\tau+1} = 0$, we have that $F_{\tau+1,P}(Q) = (F_{\tau,P}(Q))^2 \times h_1(Q)$, $[j]P = (X_j, Y_j, Z_j)$, where j is obtained by reading the τ first bits of r and $T = [2j]P = (X_{2j}, Y_{2j}, Z_{2j})$.

Using the equation of h_1, we obtain the following equality:

$$F_{\tau+1,P}(Q) = (F_{\tau,P}(Q))^2 \times$$

$$\left(Z_{2j}Z_j^2 y\sqrt{\nu} - 2Y_j^2 - 3(X_j - Z_j^2)(X_j + Z_j^2)(xZ_j^2 - X_j)\right).$$

Considering that the secret is the point P, we know j, τ, the coordinates of Q. The Miller algorithm gives us $F_{\tau+1,P}(Q)$ and $F_{\tau,P}(Q)$. We calculate the ratio $R = \frac{F_{\tau+1,P}(Q)}{(F_{\tau,P}(Q))^2}$. Using the theoretical form of R and its decomposition in the base B, by identification we can obtain, after simplification, the following system:

$$\begin{cases} Y_j Z_j^3 = \lambda_2, \\ Z_j^2(X_j^2 - Z_j^4) = \lambda_1, \\ 3X_j(X_j^2 - Z_j^4) + 2Y_j^2 = \lambda_0, \end{cases}$$

where we know the three values $\lambda_{0,1,2}$.

The resolution [19] of this non-linear system gives the following equation:

$$(\lambda_0^2 - 9\lambda_1^2)Z^{12} - (4\lambda_0\lambda_2^2 + 9\lambda_1^3)Z^6 + 4\lambda_1^4 = 0.$$

Solving the equation in Z_j, we find at most $24 = 12 \times 2 \times 1$ possible triplets (X_j, Y_j, Z_j) for the coordinates of the point $[j]P$. Once we have the coordinates of $[j]P$, to find the possible points P, we have to find j' the inverse of j modulo r, and then calculate $[j'][j]P = [j'j]P = P$. Using the elliptic curve equation, we eliminate triplets that do not lie on E. Then we just have to perform the Miller algorithm with the remaining points and compare it with the result obtained with the secret point P. So we recover the secret point P, in the case where $r_{\tau+1} = 0$. The case of $r_{\tau+1} = 1$ also leads to a non-linear system that can be solved using a Grobner basis.

Remark 12.1 We present the attack in Jacobian coordinates. As the attack is not dependent on the system of coordinates, it will be successful for other systems. In [19], the affine, projective, and Edwards coordinates are also treated. In the paper [58], the authors consider Hessian coordinates.

Remark 12.2 We describe the attack with the secret point being P. If the secret is the point Q, the attack is also valid — we just obtain an easier system to solve.

The attack against the Miller algorithm is efficient. A model of the attack was implemented in [46]. It is fair to wonder if this attack can be applied to a complete pairing. As the Weil pairing consists of two applications of the Miller algorithm, the Weil pairing is sensitive to this attack. For the Tate-like pairings (Ate, twisted Ate,...) the final exponentiation must be cancelled for the attack to be efficient. As the result of the Miller algorithm has no particular form, it seems difficult to cryptanalytically reverse the final exponentiation. As far as we know, it has not been done yet. El Mrabet cites several works in microelectronics that would give the result of the Miller algorithm during a Tate-like pairing computation: for example, the scan attack [61] or the under-voltage technique [2]. We describe in Section 12.2.3 a recent fault attack against the final exponentiation.

Attack against the *if* instruction

In [4], the authors propose a new fault model as well as an implementation of their fault attack.

The *if* skip fault model

In the Miller algorithm, the addition step is performed or not according to the bits of r. This decision is usually implemented with an *if* instruction. If an attacker is able to skip an *if* instruction, he can avoid the addition step if he wants to.

This fault model has several advantages. It can target the last iteration only of the Miller algorithm, and as a consequence only one fault injection is required to find the value $h_2(Q)$. This is better than when altering the counter value, where the attacker had to perform fault attacks until he finds the faulty result for two consecutive iterations. Then it is not as easy to develop a countermeasure against it as for an attack on the loop counter. In the latter case, it is enough to check the number of iterations that the chip executed. In the *if* skip case, the number of addition steps is highly dependant on the l value and can vary even if the security level of the parameters do not.

Recovery of $h_2(Q)$

Let $F_P(Q) = f^2 \cdot h_1(Q) \cdot h_2(Q)$ be the result of the (correct) Miller algorithm expressed with the variables of the last iteration.

If an attacker skips the *if* instruction in the last iteration, he obtains the value $F_P(Q)^* = f^2 \cdot h_1(Q)$.

With a faulty result and a correct one, he can then compute the ratio

$$\frac{F_P(Q)}{F_P(Q)^*} = \frac{f^2 \cdot h_1(Q) \cdot h_2(Q)}{f^2 \cdot h_1(Q)} = h_2(Q). \tag{12.4}$$

Finding the secret with $h_2(Q)$. With the value $h_2(Q)$, the attacker still has to find the secret (the point P in our case). The following computations are done for the Tate pairing in particular. In this case the value r is the order of the groups used in the pairing. As a consequence, in the last iteration, the equation $T = -P$ holds.

In affine coordinates in the last iteration, with an embedding degree 2, $h_2(Q) = x_Q - x_P$ since $T = -P$: the line is the vertical passing through P. So, knowing the value $h_2(Q)$, the attacker can find x_P with x_Q known. Using the elliptic curve equation, two candidates are possible for the y_P value. By trying the two possible input points in the Miller algorithm, he can find y_P with the comparison of these two Miller results and the correct one.

The result in Jacobian coordinates is slightly different. The equations are computed with an embedding degree 4 and the basis $B = \{1, \xi, \sqrt{\nu}, \xi\sqrt{\nu}\}$. The point P has Jacobian coordinates (x_P, y_P, z_P) and Q has coordinates $(x_Q, y_Q\sqrt{\nu})$.

In the last iteration, the simplified value $h_2(Q)$ is $h_2(Q) = z_P^2 x_Q - x_P$. When the attacker computes the ratio $R = \frac{F_P(Q)}{F_P(Q)^*}$, he finds a value that can be decomposed on the basis B:

$$R = R_0 + R_1\xi + R_2\sqrt{\nu} + R_3\xi\sqrt{\nu}.$$

The decomposition of $h_2(Q)$ on the basis B yields the system

$$\begin{aligned} R_1 &= z_P^2 x_{Q1} && (12.5) \\ R_2 &= z_P^2 x_{Q0} - x_P, && (12.6) \end{aligned}$$

where $x_Q = x_{Q0} + x_{Q1}\xi$.

Since Q is known to the attacker, this system can be solved to provide the values z_P^2 and then x_P. There are four possible candidates for the point P, which have to be verified by comparing them with the correct result of the Miller algorithm.

Remark 12.3 In the case of other pairings (Ate,...), the same attack can be applied. The main difference is that we find a point multiple of P: λP for a public integer λ. Indeed, we consider that except for the secret point, every detail of the implementation is public.

An implementation of the attack. The authors of this attack [4] implemented their attack on a chip, an ATmega128L, with a laser fault injection. They demonstrated the feasibility of the *if* instruction skip on a dummy algorithm mimicking the structure of the Miller algorithm. After locating the right spot for the laser fault injection, they were able to successfully skip an *if* instruction.

The *if* instruction skip has two big advantages. It easily targets a specific iteration in the Miller algorithm. It is possible to combine it with another instruction skip in the final exponentiation in order to realize a full attack on the pairing computation algorithm. But this latter possibility is yet to be proven experimentally.

Countermeasures

Several countermeasures can be implemented to prevent a fault attack. They are referenced in [22], and we briefly recall them here. We can preventively use randomization of the inputs in order to prevent any leakage of information or detect any alteration of the circuit and then abort the pairing computation.

In order to detect any alteration of the computation we can

- duplicate the computation using bilinerarity: $R_1 = e(P, Q)$, $R_2 = e(aP, bQ)$ and check if $R_2 = R_1^{ab}$ [22];
- check intermediate results during the computation: verify that the points are still on the elliptic curve, compare the last point T with $(r-1)P$ [22];
- use fault-resilient counters to avoid attacks focused on changing the Miller loop bound [43, Section 5.3];
- implement the algorithm to perform a random number of iterations greater than the correct one [24, Section 4].

The randomization and blinding methods are both based on the bilinearity of pairings:

- choose integers a and b such that $ab = 1 \mod (r)$ and compute $e(P, Q) = e(aP, bQ)$ [45];
- choose a random point R such that $S = e(P, R)^{-1}$ is defined and compute $e(P, Q) = e(P, Q + R)S$;
- use the homogeneity property of Jacobian and projective coordinates to represent the point P;
- use the homogeneity property of Jacobian and projective coordinates to represent the point Q (with a modification of the equations in the Miller algorithm);
- randomize the input points using a random field element and modify the pairing algorithm in order to cancel out the effects [53].

12.2.3 A Fault Attack against the Final Exponentiation

The main difficulty faced by fault attacks on the pairing is the final exponentiation. Even if efficient schemes are able to reverse the Miller algorithm, they still require the attacker to have access to the result of the Miller algorithm, correct or faulty.

Several possibilities have been proposed to access these values. First, for some exponents (e.g., $q^3 - 1$), it is possible to reverse the final exponentiation by using the structure of the Miller result as shown in [45]. A more implementation-dependent approach has been proposed in [19], where the authors propose to realize a scan chain attack or to completely override the final exponentiation, to directly read the result of the Miller algorithm.

Despite having been previously considered unrealistic, multiple fault injections during one execution of an algorithm seem to be more and more feasible, with some new results in this direction [55]. This new possibility opens the door to a new scheme, where two fault attacks are combined: one to reverse the final exponentiation, one to reverse the Miller algorithm.

Until recently, the final exponentiation was thought to be an efficient countermeasure against the fault attacks on the Miller algorithm, since it is mathematically impossible to find the unique preimage of the exponentiation and thus the result of the Miller loop. However, in [40], the authors propose a fault attack to reverse the final exponentiation.

Description of the attack

They chose the case where the embedding degree $k = 2d$ is even, and they attack the final exponentiation algorithm proposed in [50].

The exponent is $\frac{p^k-1}{r}$ and can be decomposed as $\frac{p^k-1}{r} = (p^d-1) \cdot \frac{p^d+1}{r}$. If the result of the Miller algorithm is noted f, we choose the following notation: $f_2 = f^{p^d-1}$ and $f_3 = f_2^{\frac{p^d+1}{r}}$ (f_3 is the pairing result observed at the end of the computation). Since $f \in \mathbb{F}_{p^k}^*$, f, f_2, and f_3 satisfy the relations

$$f^{p^k-1} = 1 \ ; \ f_2^{p^d+1} = 1 \ ; \ f_3^r = 1. \tag{12.7}$$

These relations show that these intermediary values belong to the groups noted $f_2 \in \mu_{p^d+1}$ and $f_3 \in \mu_r$.

Let $\mathbb{F}_{p^k} = \mathbb{F}_{p^d}[w]/(w^2-v)$ be the construction rule for the $\mathbb{F}_{p^k}$ extension field. v is a quadratic nonresidue in $\mathbb{F}_{p^d}$ and is a public parameter.

Let $f_2 = g_2 + h_2 \cdot w$ with $g_2, h_2 \in \mathbb{F}_{p^d}$. Then $f_2^{p^d+1} = 1$ implies $g_2^2 - v \cdot h_2^2 = 1$.

First fault

But this equation holds because $f_2 \in \mu_{p^d+1}$. If an attacker now injects a fault of value $e \in \mathbb{F}_{p^d}$ such that the faulty value f_2^* equals

$$f_2^* = f_2 + e \notin \mu_{p^d+1}, \tag{12.8}$$

it is possible to write the fault effect as

$$f_2^* = (g_2 + e) + h_2 \cdot w, \tag{12.9}$$

and the value $(f_2^*)^{p^d+1}$ can be computed by the attacker, since he can measure the value f_3^* and r:

$$(f_2^*)^{p^d+1} = (f_3^*)^{\in}\mathbb{F}_{p^d}. \tag{12.10}$$

Moreover,

$$\begin{aligned} (f_2^*)^{p^d+1} &= (g_2+e)^2 - v \cdot h_2^2 \\ &= 1 + 2 \cdot e \cdot g_2 + e^2. \end{aligned}$$

If the attacker knows the error value e, he can compute

$$g_2 = \frac{(f_3^*)^l - 1 - e^2}{2 \cdot e}, \tag{12.11}$$

and deduce the two candidates for h_2

$$h_2^+ = \sqrt{\frac{g_2^2-1}{v}} \ ; \ h_2^- = -\sqrt{\frac{g_2^2-1}{v}}. \tag{12.12}$$

With one fault, the attacker found the intermediary value f_2 by checking the two candidates and comparing $(f_2^+)^{\frac{p^d+1}{r}}$ and $(f_2^-)^{\frac{p^d+1}{r}}$ with f_3.

Second fault

At this step, the attacker knows that f_3 is the correct result of the pairing computation, and that the intermediary value is f_2. Let $f = g + h \cdot w$, $f^{-1} = g' + h' \cdot w$, and $f_2 = f^{p^d-1}$. Then we note K, the ratio

$$K = \frac{g_2 - 1}{v \cdot h_2} = \frac{h'}{g'} = -\frac{h}{g}. \tag{12.13}$$

In order to recover f, the attacker creates a new fault $e_2 \in \mathbb{F}_{p^d}$ during the inversion in the exponentiation by exponent $p^d - 1$.

Then

$$f_2 = f^{p^d-1} = \bar{f} \cdot f^{-1} \text{ and } f_2^\star = \bar{f} \cdot (f^{-1} + e_2). \tag{12.14}$$

Let Δ_{f_2} be the difference $\Delta_{f_2} = f_2^\star - f_2 = \bar{f} \cdot e_2$. Since $e_2 \in \mathbb{F}_{p^d}$, Δ_{f_2} can be written $\Delta_{f_2} = \Delta_{g_2} + \Delta_{h_2} \cdot w$ with $\Delta_{g_2} = e_2 \cdot g$ and $\Delta_{h_2} = -e_2 \cdot h$.

As $f_2^\star$ is not in μ_{p^d+1} with high probability, the attacker can compute $(f_2^\star)^{p^d+1} = (f_3^\star)^r \in \mathbb{F}_{p^d}$. Here

$$\begin{aligned}(f_3^\star)^{p^d+1} &= (g_2 + \Delta_{g_2})^2 - v \cdot (h_2 + \Delta_{h_2})^2 \\ &= (g_2 + e_2 \cdot g)^2 - v \cdot (h_2 - e_2 \cdot h)^2.\end{aligned}$$

Using the relation $h = -g \cdot K$, we obtain

$$g^2 \cdot e_2^2 \cdot (1 - v \cdot K^2) + g \cdot 2 \cdot e_2 \cdot (g_2 - v \cdot K \cdot h_2) + 1 - (f_3^\star)^r = 0. \tag{12.15}$$

This quadratic equation provides two solutions for g, each one giving only one possibility thanks to K. The attacker has two candidates for f if he knows e_2.

If he does not exactly know the fault values but is able to have a limited number of guesses, he can still find f_2 easily. But in order to find f he will have to inject more faults similar to the second one in order to uniquely determine f.

As a conclusion, with a minimum of two separate faults during two executions (plus one correct execution) of the pairing computation, the attacker is able to reverse the final exponentiation.

A notable fact about this fault attack is that it can be achieved with instruction skip faults. As a consequence, it is possible to combine it with a fault on the Miller algorithm, if the attacker can inject two faults in the same execution, in order to achieve a full-pairing fault attack.

A major disadvantage of this attack, making it easy to counter, is that the attacker must be able to observe $f_3^\star = (f_2^\star)^{\frac{p^d+1}{r}}$. But often, since $f_2 \in \mu_{p^d+1}$ is called a unitary element, it is possible to speed up the final exponentiation computation by replacing the inversions in the computation of f_3 by conjugations (which is equivalent to an inversion for unitary elements). As a consequence, the attacker cannot observe $f_3^\star$ in this case and he cannot realize the attack.

12.2.4 Conclusion

We presented the vulnerability to fault attacks of pairing algorithms when used in an identity based protocol. The first attack against Duursma and Lee algorithm targets the number of iterations. The final exponentiation in this case can be reversed using cryptanalytic equations. The most efficient pairings are constructed on the Tate model: an execution of the Miller algorithm followed by a final exponentiation. The Miller algorithm and the final exponentiation were separately submitted to fault attacks. The Miller algorithm was attacked by a modification of the number of iterations and by the corruption of the *if* condition during the last iteration. The final exponentiation was attacked using two "independent" errors in the computation.

For once, it would be interesting to validate all those fault attack schemes on practical implementations running on an embedded chip. Moreover, in order to attack a whole Tate-like pairing, further work is necessary. It would be interesting to try to attack, at the same time, the Miller algorithm and the final exponentiation. We also highlight the fact that a more general pairing constructed over an algebraic variety is sensitive to a fault attack. As a conclusion, we can say that the fault attack is a threat against an identity-based protocol, and consequently any implementation of pairings should be protected against physical attacks.

TABLE 12.1 Summary of the presented attacks.

Attack name	Target	Attack path	Fault model	Number of faults required (+ correct execution)
Page and Vercauteren [45]	Duursma and Lee algorithm	Loop counter	Data modification	$n \mid P(n,N) > 0.5$ (+1)
Whelan and Scott [59]	Miller algorithm	Sign change	Bit-flip	1 (+1)
El Mrabet [19]	Miller algorithm	Loop counter	Data modification	$n \mid P(n,N) > 0.5$ (+1)
Bae et al. [4]	Miller algorithm	If skip	Instruction skip	1 (+1)
Lashermes et al. [40]	Final exponentiation	Group change	Data modification	2+ (+1)
El Mrabet [20]	Pairing on Theta functions	Loop counter	Data modification	1

In Table 12.1 $P(n, N)$ is the probability to obtain two consecutive numbers after n picks among N integers (cf Section 12.2.2).

12.3 Countermeasures against Fault Attacks on Pairing-Based Cryptography

The protection scheme that we present here is based on the technique of *modular extension*, which was introduced by Shamir along with the first software countermeasure against fault injection attacks on CRT-RSA [52]. Joye, Paillier, and Yen noticed, two years later in [34], that the same protection could extend to any modular function. Since then, many countermeasures based on modular extension have been developed for CRT-RSA, and the method made its way to elliptic curve cryptography (ECC). In particular, Blömer, Otto, and Seifert [12], and Baek and Vasyltsov [5] applied this protection method to elliptic curve scalar multiplication (ECSM). More recently, Rauzy, Moreau, Guilley, and Najm [48] have formally studied the protection of ECSM computations with the modular extension method. We here extend it to pairing-based cryptography.

12.3.1 Modular Extension

The general idea of modular extension is to lift the computation into an over-structure (e.g., an overring) which allows us to quotient the result of the computation back to the original structure, as well as quotienting a "checksum" of the computation to a smaller structure. What has just been described (the original structure and the smaller structure) is the *direct product* of the underlying algebraic structures. If an equivalent computation is performed in parallel in the smaller structure, its result can be compared with the checksum of the main computation. If they are the same, we have a high confidence in the integrity of the main computation. This protection is sketched in Figure 12.2.

The confidence degree depends directly on the size of the smaller structure, which is thus a security parameter: the larger it is, the less probable it is to have an unwanted collision, but the more costly the redundancy will be. Indeed, the fault non-detection probability ($\mathbb{P}_{\text{n.d.}}$) is inversely proportional to the size of the small structure.

When the basic structure underlying the original computation is a field, as is the case in pairing-based cryptography (contrary to, e.g., RSA, which only requires a ring), a problem arises with inversions. Indeed, if we call $\mathbb{F}_p$ the original structure and $\mathbb{F}_r$ the smaller one, the nonzero elements of their direct product $\mathbb{Z}_{pr}$ do not all have an inverse. Nonetheless, this problem can be circumvented.

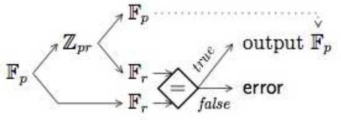

FIGURE 12.2 Sketch of the principle of *modular extension*.

PROPOSITION 12.1 *To get the inverse of z in $\mathbb{F}_p$ while computing in $\mathbb{Z}_{pr}$, one has:*

- $z = 0 \bmod r \implies (z^{p-2} \bmod pr) \equiv z^{-1} \mod p$,
- *otherwise* $(z^{-1} \bmod pr) \equiv z^{-1} \mod p$.

Remark 12.4 Golić and Tymen introduce in [25] a masking countermeasure of the Advanced Encryption Standard (AES [1]), called the "embedded multiplicative masking", which also requires embedding a finite field into a larger ring. In this context, the over-structure is a *polynomial extension* of some extension of $\mathbb{F}_2$, but the idea is similar to *modular extension*. In particular, the authors note in Section 5.1 of their paper [25] that inversion in the base field can be obtained in the overring as an exponentiation to the base field order minus two.

But the inversion procedure we give in Proposition 12.1 is novel, in that we allow an optimization if the number is inversible in the overring. This requires a test, which we can do safely without disclosing information in the context of fault-attacks detection. Nonetheless, such optimization would be insecure in the context of the "embedded multiplicative masking" countermeasure, since this would leak information about the value of the mask. This is a first-order flaw that would undermine the security of the "embedded multiplicative masking" protection against side-channel attacks.

In addition, it is possible to write pairing algorithms that use very few divisions (as little as a single one in our mini-pairing implementation; see hereafter in Section 12.3.3).

12.3.2 Other Existing Countermeasures

We review the three known methods to apply and/or adapt the modular extension countermeasure to ECSM (which is central to pairing-based cryptography).

In [12], Blömer, Otto, and Seifert (BOS) suggest applying the modular extension countermeasure by replacing finite fields and rings with elliptic curves over finite fields and rings. Let us denote the nominal elliptic curve as $E(\mathbb{F}_p)$. Then the protection by BOS consists in achieving the same computation, but on a larger elliptic curve $E(\mathbb{Z}_{pr})$, and on a small elliptic curve $E(\mathbb{F}_r)$. According to the authors, the reduction of the result of the ECSM on $E(\mathbb{Z}_{pr})$ modulo r should yield exactly the result of the ECSM on $E(\mathbb{F}_r)$. If not, then an error is suspected, otherwise the result of the ECSM on $E(\mathbb{Z}_{pr})$ is reduced modulo p, which should be the correct result. The rationale of BOS is illustrated in Figure 12.3. Apart from the lacunar management of inversions in $\mathbb{Z}_{pr}$, one other caveat is pinpointed in [48, § 3.1]. Due to the existence of unrelated tests (e.g., equality of intermediate points to the point at infinity) on $E(\mathbb{Z}_{pr})$ and $E(\mathbb{F}_r)$, the algorithm proposed by BOS is incorrect, meaning that it can return an error when there has been none. These *false positives* are harmful in that they leak information on the scalar.

In [5], Baek and Vasyltsov (BV) present an optimization of BOS. The idea is to avoid the computation on $E(\mathbb{F}_r)$, but to trade it for a verification that the ECSM result on $E(\mathbb{Z}_{pr})$ modulo r belongs to $E(\mathbb{F}_r)$, i.e., that it satisfies its Weierstrass equation taken modulo r. The rationale of BV is illustrated in Figure 12.4: notice that in this figure, the security parameter r is chosen to be a prime. This was not mandated in the original BV publication, but shall definitely be preferred for the countermeasure to have reasonable detection probability. The BV protection is more efficient than BOS, since the verification of BV is, computationally speaking, easier than an ECSM on $E(\mathbb{F}_r)$ (even if the scalar is reduced modulo the order of the small curve $E(\mathbb{F}_r)$). Besides, to avoid dealing with inversions in $\mathbb{Z}_{pr}$, BV is executed in projective coordinates, the projective-to-affine conversion only being carried out after the integrity verification. Still, BV runs into the problem of inconsistent tests before elliptic curves point addition and doubling. The consequence is that, depending on the scalar (and the fixed generator point), the "virtual"

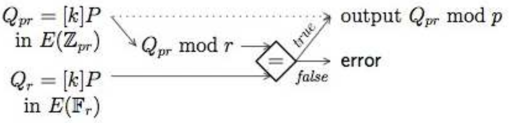

FIGURE 12.3 Principle of protection of ECSM against fault attacks by BOS [12].

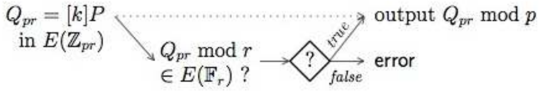

FIGURE 12.4 Principle of protection of ECSM against fault attacks by BV [5].

computation on $E(\mathbb{F}_r)$ (the modulo r of computation on the embedding elliptic curve $E(\mathbb{Z}_{pr})$) can be stuck at the point at infinity.

In [48], Rauzy, Moreau, Guilley, and Najm (RMGN) notice that the tests' inconsistencies on $E(\mathbb{F}_r)$ can also be security weaknesses. Indeed, when the mirror computation on $E(\mathbb{F}_r)$ is stuck at the point at infinity, most faults (for instance, faults touching only one of three projective coordinates) are undetected, because the computation naturally brings the intermediate point at the point at infinity on $E(\mathbb{F}_r)$ (and to a point with coordinates that are null modulo r on $E(\mathbb{Z}_{pr})$). Thus, the probability of fault non-detection is increased with respect to the expected $O(1/r)$. Consequently, RMGN propose a straightforward application of the modular extension method (as suggested by Joye, Paillier, and Yen [34]) to ECSM, where all tests on points are simply removed. From a functional point of view, this does not raise an issue, as in practice scalars are chosen to be smaller than the base point order, so that tests can be safely skipped. The pro is that this method is correct (it has no false positives), but the con is that some faults are undetected (the behavior is identical to that of BV). Indeed, even though in RMGN there is no notion of elliptic curve $E(\mathbb{F}_r)$, the values in $\mathbb{F}_r$ can be stuck at 0 (though we can still detect those faulty cases beforehand by comparing the order of $\mathbb{F}_r$ with the scalar). However, as in the case of BV, the increase of fault non-detection probability is limited, and can be tolerated with large enough values of r (e.g., 32-bit values). Indeed, as detailed in [48, Proposition 7 in Sec. 6.3], the probability of fault non-detection remains $O(\frac{1}{r})$.

12.3.3 Walk through Mini-Pairing

As an example of the *modular extension* protection scheme that we present here, we provide both an unprotected and a protected implementations of the optimal Ate pairing that we call "mini-pairing".* The provided code has been implemented in C using the GMP big number library, more precisely `mini-gmp`, a portable version of GMP with a reduced number of functions. The parameters of the optimal Ate pairing we used are presented in Figure 12.5.

The protected version of the optimal Ate pairing is given in Algorithm 12.5.

Here we discuss the differences between the unprotected and the protected mini-pairing implementations. Indeed, for the sake of simplicity, we emphasize our comments on the necessary code modifications to implement the modular extension protection scheme, rather than focusing on the underlying algorithm, namely an optimal Ate pairing. For the same reasons, the implementation has not been optimized.

*The code is available here: `http://pablo.rauzy.name/research/sources/hopbc_mini-pairing.tgz`.

Field characteristic	$p =$	0x2523648240000001ba344d8000000008612100000000013a7000000000000013
Curve equation coefficients	$a =$	0x0
	$b =$	0x2
Points coordinates	$Qx =$	0x1c2141648fed8ba0f2a3febe8b98509bf86398d1fd1050c88fc3d88be3d15db1 + 0x93772aa06ea4acf488ce113f4a56aeb6c23264001c1501c1c59cd47faac6d0f$\cdot u$
	$Qy =$	0x1eb672f0d5335990c9b12f9839b1a8804393211b198237c5acfc4d69d51186a0 + 0x118c5d037558e51efdd3cf3530d8c5cb65c52f9cf639ed6d81ddc6c16b76eec0$\cdot u$
	$Px =$	0x2523648240000001ba344d8000000008612100000000013a7000000000000012
	$Py =$	0x1

FIGURE 12.5 Parameters of mini-pairing.

ALGORITHM 12.5 Optimal Ate pairing capable of detecting faults (using entanglement strategy).

```
let $p$,$r$ be two primes, and $\mathbb{F}_p$, $\mathbb{F}_r$ two fields with $p$ and $r$ elements
let $\mathbb{Z}_{pr}$ be the direct product ofq $\mathbb{F}_p$ and $\mathbb{F}_r$
let $G_1$,$G_2$ be two additive cyclic groups of prime order $p$
let $e$ be the pairing mapping $G_1$ and $G_2$
let $P \in G_1$ and $Q \in G_2$
compute $e_{\mathbb{Z}_{pr}} = e(P,Q)$ in $\mathbb{Z}_{pr}$
compute $e_{\mathbb{F}_r} = e(P,Q)$ in $\mathbb{F}_r$
if $e_{\mathbb{F}_r} = e_{\mathbb{Z}_{pr}} \bmod r$ then
    return $e(P,Q) = e_{\mathbb{Z}_{pr}} \bmod p$
else
    return error
end
```

The first modification is obviously the addition of variables that store newly needed values, such as the security parameter r (lines 1314 and 1315 at the beginning of the `main` function in `mini-pairing_protected.c`). Then, the main change induced by the protection is that the pairing algorithm is now called twice: once in $\mathbb{Z}_{pr}$, and once in $\mathbb{F}_r$. Following these computations, we need to check whether the redundancy invariant held, i.e., to test whether both outputs are equal modulo r. Two additional functions are defined for this purpose: one to compare two elements of $\mathbb{F}_{r^{12}}$ (`p12_is_eq`, lines 1241 to 1308 of `mini-pairing_protected.c`), and another to cast an element from $\mathbb{Z}_{(pr)^{12}}$ to $\mathbb{F}_{r^{12}}$ (`p12in`, lines 1180 to 1239 of `mini-pairing_protected.c`). The redundancy check and error display (if need be) are then performed at line 1365.

Another difference is in the inversion (lines 316 to 324 of `mini-pairing_protected.c`). Inversions seldom occur in this pairing algorithm; however, it will fail if the input number is a multiple of r in $\mathbb{Z}_{pr}$. In such a case, we simulate an inversion in $\mathbb{F}_p$ (which is what we actually need) by exponentiating to $p-2$, as explained in Proposition 12.1. Computing an exponentiation is more costly than computing an inversion with an extended Euclidean algorithm. But there are few enough occurrences of this case in practice that this workaround does not have a significant impact on the execution time.

12.3.4 Overhead

Here we present the cost of the countermeasure as deployed in our mini-pairing implementation. Note that for the sake of simplicity and clarity, the implementation is not optimized and is thus quite slow. However, the overhead factor is still relevant, since optimizations of the pairing algorithm would directly benefit the protected version as we constructed it (see Section 12.3.3).

TABLE 12.2 Performance for mini-pairing on an ARM Cortex-M4 μc.

$\mathbb{F}_p$ time (ms) Miller's loop	intermediate computations	final exponentiation	sum
4160	621	6519	11343

TABLE 12.3 Modular extension performance for mini-pairing on an ARM Cortex-M4 μc.

r size	$\mathbb{Z}_{pr}$ time (ms)				$\mathbb{F}_r$ time (ms)				total	over-
(bit)	ML	IC	FE	sum	ML	IC	FE	sum	(ms)	head
8	4576	703	7201	12443	1186	142	1781	3105	15587	×1.37
16	4617	706	7263	12546	1185	141	1777	3097	15685	×1.38
32	4706	725	7407	12864	1042	126	1565	2726	15590	×1.37
64	5260	834	8302	14334	1370	171	2071	3618	17984	×1.59

ML = Miller's loop, IC = intermediate computations, FE = final exponentiation.

Speed

Times are measured on an ARM Cortex-M4 microcontroller. Table 12.2 gives the timing of the unprotected implementation that serves as reference to compute the overheads given in Table 12.3. Table 12.3 presents the cost of the countermeasure for different sizes of the security parameter r, using the largest prime number of each size.

Table 12.3 shows the good performance results of the modular extension protection scheme. We can see that when r is on 32 bits, the alignment with `int` makes `mini-gmp` faster, incurring a factor of only ≈ 1.37 in the total run time compared to the unprotected algorithm, similar to the cost with r on 8 bits but with a much higher resistance.

Space

Table 12.4 shows the cost of the countermeasure in terms of code size, both for the programmer (in number of lines of C code), and for the hardware (in kilobytes of executable code and in bytes of occupied memory). Note that the executable code size also accounts for embedded libraries such as `mini-gmp`.

In order to measure the RAM usage, the maximal value of the heap pointer address is monitored. This is achieved by equipping the `_sbrk()` function, located in `syscall.c`, which is called by `malloc()` and `free()`. Notice that most of the RAM is indeed used by the heap (and not the stack), because in the ECSM code, all parameters are passed by address, and there are neither recursive functions nor pre-initialized tables (but for the elliptic curve parameters).

As expected, we can see in Table 12.4 that the implementation of the modular extension countermeasure is cheap in terms of engineering: less than 150 additional lines of code (for a total of almost 1400 lines); as well as in terms of resources: the executable code is only marginally larger, and memory usage is essentially the same (probably due to the way `mini-gmp`'s and libC memory allocation works).

TABLE 12.4 Modular extension cost in terms of space for mini-pairing on an ARM Cortex-M4 μc.

implementation	code size (LoC)	executable size (B)	occupied RAM (kB)
unprotected	1404	95032	≈ 20
protected	1545 (+141)	96832 (+1800)	≈ 20

12.3.5 Security Evaluation

DEFINITION 12.1 (Fault model) We consider an attacker who is able to fault data by randomizing or zeroing any intermediate variable, and fault code by skipping any number of consecutive instructions.

DEFINITION 12.2 (Attack order) We call order of the attack the number of faults (in the sense of Definition 12.1) injected during the target execution.

Remark 12.5 In the rest of this section, we focus on the resistance to first-order attacks on data. Indeed, Rauzy and Guilley have shown in [47] that it is possible to adapt the modular extension protection scheme to resist attacks of order D for any D by chaining D repetitions of the final check in a way that forces each repetition of the modular extension invariant verification to be faulted independently, and faults on the code can be formally captured (simulated) by faults on intermediate variables.

The security provided by the modular extension protection scheme has been formally studied in [48, § 5]. Although the practical study was carried out on an ECSM algorithm, the theoretical results are still valid in the context of pairing algorithms (or actually any other modular arithmetic computations): the probability of not detecting a fault $\mathbb{P}_{\text{n.d.}}$ is inversely proportional to the security parameter r, i.e., $\mathbb{P}_{\text{n.d.}} = O(\frac{1}{r})$.

Indeed, we consider that a fault might be exploitable as soon as the algorithm outputs a value that is different from the expected result in absence of faults. In the modular extension setting, this can happen if and only if the result of the computation in $\mathbb{Z}_{(pr)^{12}}$ is equal to the result of the computation in $\mathbb{F}_{r^{12}}$ modulo r, while being different from the expected result modulo p. The probability of this happening is $\frac{1}{r}$ if we consider that values in $\mathbb{F}_r$ are uniformly distributed, which is quite reasonable given that $r \ll p$. As a matter of fact, we can quantify this distribution. Let U uniformly distributed in $\{0, \ldots, p-1\}$, then $V = U \mod r$ has a piecewise constant distribution. Let v in $\{0, \ldots, r-1\}$, we have:

$$\mathbb{P}(V = v) = \begin{cases} \frac{1}{p}(\lfloor \frac{p}{r} \rfloor + 1) & \text{if } v < (p \bmod r) \\ \frac{1}{p}(\lfloor \frac{p}{r} \rfloor) & \text{otherwise.} \end{cases}$$

There are other vulnerabilities, but they do not alter $\mathbb{P}_{\text{n.d.}}$. For instance, the final exponentiation always returns 0 in $\mathbb{F}_{r^{12}}$ for some (small) values of r. In addition, Miller's algorithm manipulates an element from an elliptic curve on $\mathbb{F}_{p^2}$, and if a fault manages to set the Y coordinate of that element to 0 mod r, its other coordinates will also become multiples of r after few iterations of the Miller's loop, thereby "infecting" the computation by being completely equal to 0 modulo r. Therefore, the exponentiation will also output 0 modulo r in $\mathbb{Z}_{(pr)^{12}}$, and the final test won't detect the fault. However, such faults are highly unlikely in practice, the probability being roughly $\frac{1}{r^2}$, which is why $\mathbb{P}_{\text{n.d.}}$ stays $O(\frac{1}{r})$. Anyway, it is advised to use large enough values for the security parameter r. In practice, 32-bits values are recommended as they are large enough to offer a good security while not being big enough for the overhead induced by the countermeasure to be prohibitive (see Table 12.3). It is also advised to use prime numbers for r as it will diminish the probability of occurrence of the inversion problem mentioned above.

Remark 12.6 One must also be careful with the choice of parameters. For instance, manipulating a P whose coordinates are multiples of r might lead to singularities in the computation in $\mathbb{F}_r$, singularities such as the $\mathbb{F}_{r^{12}}$-output being equal to 1, therefore making the pairing computation more vulnerable to fault injections.

Remark 12.7 Taking the BV fault detection as an example (recall Figure 12.4), one might be tempted to lift the computation from $\mathbb{F}_r$ to $\mathbb{Z}_{pr}$, and do a sanity check of the pairing computation instead of redoing a redundant pairing computation in $\mathbb{F}_r$. One property that could be checked is the bilinearity. Unfortunately, the elliptic curve changes modulo r, as the Weierstrass coefficients are reduced modulo r. Therefore, the bilinearity remarkable identity is not preserved in $\mathbb{F}_r$ after reducing the pairing computation from $\mathbb{Z}_{pr}$ modulo r.

The presented countermeasure can also bring a reasonable security against simple side-channel attacks. Indeed, the 32-bits r parameter can be chosen randomly, and there are 98182657 prime numbers between 2^{31} and 2^{32}, hence providing many different possible execution traces against power analysis.

12.4 Countermeasures against Side-Channel Attacks on Pairing-Based Cryptography

In order to protect pairing implementations against the side-channel attacks described in this chapter, several countermeasures have been proposed. The aim of most of those countermeasures is to avoid any predictable link between the manipulated data and the known input.

In practice, in the pairing computation context, there are different randomization levels. One category of countermeasures consists of randomizing the inputs before the pairing computation. Another one consists of adding a random mask directly into the Miller algorithm. Moreover, a method based on arithmetic randomization can be adapted for the pairing.

12.4.1 Randomization of the Inputs

Page and Vercauteren [44] proposed two countermeasures for their passive attack. The first one is based on the pairing bilinearity. Let a and b be two random values, then $e([a]P,[b]Q)^{\frac{1}{ab}} = e(P,Q)$. For each pairing computation, one can thus take different a and b and compute $e([a]P,[b]Q)^{\frac{1}{ab}}$. This method is clearly very costly in terms of computation time. Then, the random choice for a and b can be adapted to have $a = b^{-1} \mod q$, so the exponent $\frac{1}{ab}$ is equal to 1.

The same authors propose another method, for instance in the case where P is secret, consisting of adding the mask to the point Q in the following way: select a random point $R \in \mathbb{G}_2$ and compute $e(P,Q+R)e(P,R)^{-1}$ instead of $e(P,Q)$, with different values of R at every call to e.

Widely inspired by the previous protection, Blömer et al. in [11] proposed an improvement applied for the Tate pairing. In the reduced Tate pairing, they note that the set of the second argument input is the equivalence class $\frac{E(\mathbb{F}_{p^k})}{rE(\mathbb{F}_{p^k})}$. They hence choose a random point $R \in E(\mathbb{F}_{p^k})$ with order l and coprime to r. Then $Q+R \sim Q$. Hence $e(P,Q+R) = e(P,Q)$. This method avoids the second pairing computation that is used to find the same result without mask.

12.4.2 Randomization of Intermediate Variables

Kim et al. [37] use the third countermeasure proposed by Coron in [15], using random projective coordinates to protect the Eta pairing in characteristic 2. But it can be adapted to pairing algorithms based over a large prime characteristic field. At the beginning of the algorithm, they proceed with this randomization based on the homogeneity of projective or Jacobian coordinates. For non-zero integer λ, the point $P = (X_P, Y_P, Z_P)$ in projective coordinates is also the point $P = (\lambda X_P, \lambda Y_P, \lambda Z_P)$. The point $P = (X_P, Y_P, Z_P)$ in Jacobian coordinates is also the point

$P = (\lambda^2 X_P, \lambda^3 Y_P, \lambda Z_P)$.

12.4.3 Arithmetic Randomization

However, all previous attacks against pairings targeted an arithmetic operation. Securing multiplications was originally studied in [30] in order to protect ECDSA against side-channel attacks. The aim is to avoid all possible predictions during a modular multiplication. A mask is randomly chosen before processing a multiplication. Then it is impossible to make any hypothesis on the output of internal modular multiplication. We find another masking technique in the paper [9], the aim being the same: avoiding any predictable link between known and secret data directly in the arithmetic.

Protected arithmetic can also be obtained with the well-known Residue Number System method [7].

Arithmetic protection seems to be a robust method against side-channel. However, it is necessary to evaluate the overhead cost. Indeed, changing permutation in randomized multiplication or refreshing RNS basis in case of RNS implementation have a significant overhead.

12.4.4 Countermeasures against Loop Reduction

The fault attacks against the Miller algorithm rely on the modification of the number of iterations performed by the algorithm. We can add a counter to the Miller algorithm.

12.4.5 Pseudo Code of Unified Existing Countermeasures

A set of these protections is relatively easy to implement. Algorithm 12.6 shows a possible combination of existing countermeasures. The arithmetic randomization is directly implemented in the arithmetic. For example, the multiplication of two long integers in $\mathbb{F}_q$ can be realized by Algorithm 2 of [9] instead of classic long integer multiplication.

ALGORITHM 12.6 Computation of pairing using Miller's loop.

```
Input  : $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ with $Q$ secret, $r = (r_{w-1} \dots r_0)_2$ radix 2 representation
Output : $e(P, Q)$
Randomly pick $a$ and $b$ in $\{1, \dots, q-1\}$ such that $a = b^{-1} \mod q$
Set $P' \leftarrow [a]P$ and $Q' \leftarrow [b]Q$          // Randomization of the inputs
Randomly pick $\lambda \in \mathbb{F}'_q$
Set $T \leftarrow (\lambda x_{P'}, \lambda y_{P'}, \lambda)$   // Randomized projective coordinates
$f \leftarrow 1$
for $i = w - 2$ downto 0 do
    $f \leftarrow f^2 \cdot l_{T,T}(Q')$
    $T \leftarrow [2]T$
    if $r_i == 1$ then
        $f \leftarrow f \cdot l_{T,P'}(Q')$
        $T \leftarrow T + P'$
    end
end
return $f^{\frac{q^k-1}{r}}$
```

References

[1] FIPS PUB 197. *Advanced Encryption Standard (AES)*. National Institute of Standards and Technology, U.S. Department of Commerce, November 2001.

[2] Ross Anderson and Markus Kuhn. Tamper resistance—a cautionary note. In *Proceedings of the Second USENIX Workshop on Electronic Commerce*, pp. 1–11. USENIX Association, 1996.

[3] Diego F. Aranha, Jean-Luc Beuchat, Jérémie Detrey, and Nicolas Estibals. Optimal Eta pairing on supersingular genus-2 binary hyperelliptic curves. In O. Dunkelman, editor, *Topics in Cryptology – CT-RSA 2012*, volume 7178 of *Lecture Notes in Computer Science*, pp. 98–115. Springer, Heidelberg, 2012.

[4] Kiseok Bae, Sangjae Moon, and Jaecheol Ha. Instruction fault attack on the Miller algorithm in a pairing-based cryptosystem. In *7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2013)*, pp. 167–174. IEEE Press, 2013.

[5] Yoo-Jin Baek and Ihor Vasyltsov. How to prevent DPA and fault attack in a unified way for ECC scalar multiplication - Ring extension method. In E. Dawson and D. S. Wong, editors, *Information Security Practice and Experience (ISPEC 2007)*, volume 3439 of *Lecture Notes in Computer Science*, pp. 225–237. Springer, 2007.

[6] Jean-Claude Bajard and Nadia El Mrabet. Pairing in cryptography: an arithmetic point of view. In F. T. Luk, editor, *Advanced Signal Processing Algorithms, Architectures, and Implementations XVII*, volume 6697 of *Proc. SPIE*. SPIE, 2007.

[7] Jean-Claude Bajard, Laurent Imbert, Pierre-Yvan Liardet, and Yannick Teglia. Leak resistant arithmetic. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pp. 62–75. Springer, Heidelberg, 2004.

[8] Paulo S. L. M. Barreto, Steven Galbraith, Colm Ó hÉigeartaigh, and Michael Scott. Efficient pairing computation on supersingular Abelian varieties. *Designs, Codes and Cryptography*, 42(3):239–271, 2007.

[9] Aurélie Bauer, Eliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal and vertical side-channel attacks against secure RSA implementations. *CT-RSA*, pp. 1–17, 2013.

[10] Daniel J. Bernstein. Cache-timing attacks on AES. Unpublished manuscript, available at `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf`, 2005.

[11] Johannes Blömer, Peter Günther, and Gennadij Liske. Improved side-channel attacks on pairing based cryptography. In E. Prouff, editor, *Constructive Side-Channel Analysis and Secure Design (COSADE 2013)*, volume 7864 of *Lecture Notes in Computer Science*, pp. 154–168. Springer, 2013.

[12] Johannes Blömer, Martin Otto, and Jean-Pierre Seifert. Sign change fault attacks on elliptic curve cryptosystems. In L. Breveglieri et al., editors, *Fault Diagnosis and Tolerance in Cryptography (FDTC 2006)*, volume 4236 of *Lecture Notes in Computer Science*, pp. 154–168. Springer, 2006.

[13] Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[14] Benoît Chevallier-Mames, Mathieu Ciet, and Marc Joye. Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. *IEEE Transactions on Computers*, 53(6):760–768, 2004.

[15] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Çetin Kaya. Koç and C. Paar, editors, *Cryptographic Hardware*

and Embedded Systems – CHES'99, volume 1717 of *Lecture Notes in Computer Science*, pp. 292–302. Springer, Heidelberg, 1999.

[16] Elke De Mulder, Siddika B. Örs, Bart Preneel, and Ingrid Verbauwhede. Differential power and electromagnetic attacks on a FPGA implementation of elliptic curve cryptosystems. *Computers & Electrical Engineering*, 33(5/6):367–382, 2007.

[17] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria. Electromagnetic transient faults injection on a hardware and a software implementation of AES. In G. Bertoni and B. Gierlichs, editors, *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2012)*, pp. 7–15. IEEE Computer Society, 2012.

[18] Iwan Duursma and Hyang-Sook Lee. Tate-pairing implementations for tripartite key agreement. Cryptology ePrint Archive, Report 2003/053, 2003. `http://eprint.iacr.org/2003/053`.

[19] Nadia El Mrabet. Fault attack against Miller's algorithm. Cryptology ePrint Archive, Report 2011/709, 2011. `http://eprint.iacr.org/2011/709`.

[20] Nadia El Mrabet. Side-channel attacks against pairing over Theta functions. In T. Muntean, D. Poulakis, and R. Rolland, editors, *Algebraic Informatics (CAI 2013)*, volume 8080 of *Lecture Notes in Computer Science*, pp. 132–146. Springer, 2013.

[21] Nadia El Mrabet, Giorgio Di Natale, and Marie Lise Flottes. A practical differential power analysis attack against the Miller algorithm. In *2009 Conference on Ph.D. Research in Microelectronics and Electronics (PRIME 2009)*, pp. 308–311. IEEE Press, 2009.

[22] Nadia El Mrabet, Dan Page, and Frederik Vercauteren. Fault attacks on pairing-based cryptography. In M. Joye and M. Tunstall, editors, *Fault Analysis in Cryptography*, Information Security and Cryptography, pp. 221–236. Springer, 2012.

[23] Santosh Ghosh and Dipanwita Roy Chowdhury. Security of prime field pairing cryptoprocessor against differential power attack. In M. Joye, D. Mukhopadhyay, and M. Tunstall, editors, *Security Aspects in Information Technology (InfoSecHiComNet 2011)*, volume 7011 of *Lecture Notes in Computer Science*, pp. 16–29. Springer, 2011.

[24] Santosh Ghosh, Debdeep Mukhopadhyay, and Dipanwita Roy Chowdhury. Fault attack, countermeasures on pairing based cryptography. *International Journal of Network Security*, 12(1):21–28, 2011.

[25] Jovan Dj. Golic and Christophe Tymen. Multiplicative masking and power analysis of AES. In B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pp. 198–212. Springer, Heidelberg, 2003.

[26] Gurleen Grewal, Reza Azarderakhsh, Patrick Longa, Shi Hu, and David Jao. Efficient implementation of bilinear pairings on ARM processors. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography – SAC 2012*, volume 7707 of *Lecture Notes in Computer Science*, pp. 149–165. Springer, Heidelberg, 2013.

[27] D. H. Habing. The use of lasers to simulate radiation-induced transients in semiconductor devices and circuits. *IEEE Transactions on Nuclear Science*, 12(5):91–100, 1965.

[28] Florian Hess. Pairing lattices (invited talk). In S. D. Galbraith and K. G. Paterson, editors, *Pairing-Based Cryptography – Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pp. 18–38. Springer, Heidelberg, 2008.

[29] Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The Eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, 2006.

[30] Michael Hutter, Marcel Medwed, Daniel Hein, and Johannes Wolkerstorfer. Attacking ECDSA-Enabled RFID devices. *Applied Cryptography and Network Security*, pp. 519–534, 2009.

[31] Antoine Joux. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in small characteristic. In T. Lange, K. Lauter, and P. Lisonek, editors, *Selected Areas in Cryptography – SAC 2013*, volume 8282 of *Lecture Notes in Computer Science*, pp. 355–379. Springer, Heidelberg, 2014.

[32] Marc Joye. Elliptic curves and side-channel analysis. *ST Journal of System Research*, 4(1):17–21, 2003.

[33] Marc Joye and Gregory Neven, editors. *Identity-Based Cryptography*, volume 2 of *Cryptology and Information Security Series*. IOS press, 2009.

[34] Marc Joye, Pascal Paillier, and Sung-Ming Yen. Secure evaluation of modular functions. In R. J. Hwang and C. K. Wu, editors, *2001 International Workshop on Cryptology and Network Security*, pp. 227–229, 2001.

[35] Marc Joye and Sung-Ming Yen. The Montgomery powering ladder. In B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pp. 291–302. Springer, Heidelberg, 2003.

[36] Chong Hee Kim and Jean-Jacques Quisquater. Faults, injection methods, and fault attacks. *Design Test of Computers, IEEE*, 24(6):544–545, 2007.

[37] Tae Hyun Kim, Tsuyoshi Takagi, Dong-Guk Han, Ho Won Kim, and Jongin Lim. *Cryptology and Network Security: 5th International Conference, CANS 2006, Suzhou, China, December 8-10, 2006. Proceedings*, chapter Side-Channel Attacks and Countermeasures on Pairing Based Cryptosystems over Binary Fields, pp. 168–181. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[38] Neal Koblitz and Alfred Menezes. Pairing-based cryptography at high security levels (invited paper). In N. P. Smart, editor, *Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pp. 13–36. Springer, Heidelberg, 2005.

[39] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In M. J. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pp. 388–397. Springer, Heidelberg, 1999.

[40] Ronan Lashermes, Jacques Fournier, and Louis Goubin. Inverting the final exponentiation of Tate pairings on ordinary elliptic curves using faults. In G. Bertoni and J.-S. Coron, editors, *Cryptographic Hardware and Embedded Systems – CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pp. 365–382. Springer, Heidelberg, 2013.

[41] Rudolf Lidl and Harald Niederreiter. *Finite Fields*, volume 20 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 2nd edition, 1997.

[42] David Lubicz and Damien Robert. Efficient pairing computation with theta functions. In G. Hanrot, F. Morain, and E. Thomé, editors, *Algorithmic Number Theory (ANTS-IX)*, volume 6197 of *Lecture Notes in Computer Science*, pp. 251–269. Springer, 2010.

[43] Erdinç Öztürk, Gunnar Gaubatz, and Berk Sunar. Tate pairing with strong fault resiliency. In L. Breveglieri et al., editors, *Fourth Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2007)*, pp. 103–111. IEEE Computer Society, 2007.

[44] Dan Page and Frederik Vercauteren. Fault and side-channel attacks on pairing based cryptography. Cryptology ePrint Archive, Report 2004/283, 2004. `http://eprint.iacr.org/2004/283`.

[45] Dan Page and Frederik Vercauteren. A fault attack on pairing-based cryptography. *IEEE Transactions on Computers*, 55(9):1075–1080, 2006.

[46] Jea-Hoon Park, Gyo Yong-Sohn, and Sang-Jae Moon. A simplifying method of fault attacks on pairing computations. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 94(6):1473–1475, 2011.

[47] Pablo Rauzy and Sylvain Guilley. Countermeasures against high-order fault-injection attacks on CRT-RSA. In A. Tria and D. Choi, editors, *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2014)*, pp. 68–82. IEEE Computer Society, 2014.

[48] Pablo Rauzy, Martin Moreau, Sylvain Guilley, and Zakaria Najm. Using modular extension to provably protect ECC against fault attacks. Cryptology ePrint Archive, Report 2015/882, 2015. `http://eprint.iacr.org/2015/882`.

[49] Michael Scott. On the efficient implementation of pairing-based protocols. In L. Chen, editor, *Cryptography and Coding*, volume 7089 of *Lecture Notes in Computer Science*, pp. 296–308. Springer, Heidelberg, 2011.

[50] Michael Scott, Naomi Benger, Manuel Charlemagne, Luis J. Dominguez Perez, and Ezekiel J. Kachisa. On the final exponentiation for calculating pairings on ordinary elliptic curves. In H. Shacham and B. Waters, editors, *Pairing-Based Cryptography – Pairing 2009*, volume 5671 of *Lecture Notes in Computer Science*, pp. 78–88. Springer, Heidelberg, 2009.

[51] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pp. 47–53. Springer, Heidelberg, 1984.

[52] Adi Shamir. Method and apparatus for protecting public key schemes from timing and fault attacks. US Patent # 5,991,415, 1999. Presented at the rump session of EUROCRYPT '97.

[53] Masaaki Shirase, Tsuyoshi Takagi, and Eiji Okamoto. An efficient countermeasure against side-channel attacks for pairing computation. In L. Chen, Y. Mu, and W. Susilo, editors, *Information Security Practice and Experience (ISPEC 2008)*, volume 4991 of *Lecture Notes in Computer Science*, pp. 290–303. Springer, 2008.

[54] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, 2nd edition, 2009.

[55] Elena Trichina and Roman Korkikyan. Multi fault laser attacks on protected CRT-RSA. In L. Breveglieri et al., editors, *2010 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2010)*, pp. 75–86. IEEE Computer Society, 2010.

[56] Thomas Unterluggauer and Erich Wenger. Practical attack on bilinear pairings to disclose the secrets of embedded devices. In *9th International Conference on Availability, Reliability and Security (ARES 2014)*, pp. 69–77. IEEE Computer Society, 2014.

[57] Frederik Vercauteren. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1):455–461, 2009.

[58] Jiang Weng, Yunqi Dou, and Chuangui Ma. Fault attacks against the Miller algorithm in Hessian coordinates. In C.-K. Wu, M. Yung, and D. Lin, editors, *Information Security and Cryptology (Inscrypt 2011)*, volume 7537 of *Lecture Notes in Computer Science*, pp. 102–112. Springer, 2012.

[59] Claire Whelan and Michael Scott. The importance of the final exponentiation in pairings when considering fault attacks. In T. Takagi et al., editors, *Pairing-Based Cryptography – Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pp. 225–246. Springer, Heidelberg, 2007.

[60] Claire Whelan and Mike Scott. Side-channel analysis of practical pairing implementations: Which path is more secure? In P. Q. Nguyen, editor, *Progress in Cryptology – VIETCRYPT 2006*, volume 4341 of *Lecture Notes in Computer Science*, pp. 99–114. Springer, Heidelberg, 2006.

[61] Bo Yang, Kaijie Wu, and Ramesh Karri. Scan based side-channel attack on dedicated hardware implementations of data encryption standard. In *International Test Conference (ITC 2004)*, pp. 339–344. IEEE Press, 2004.