New Versions of Miller-loop Secured against Side-Channel Attacks

Nadia El Mrabet, Loubna Ghammam, Nicolas Méloni, Emmanuel Fouotsa

New Versions of Miller-loop Secured against Side-Channel Attacks

Nadia El Mrabet[1][0000−0003−3840−584X], Loubna Ghammam[2][0000−0003−3438−1860], Nicolas Meloni[3][0000−0001−6286−6756], and Emmanuel Fouotsa[0000−0002−3015−2780]

[1] Mines Saint-Etienne, CEA-LETI, Centre CMP, Departement SAS, F - 13541 Gardanne France
nadia.el-mrabet@emse.fr
[2] ITK Engineering GmbH, Im Speyerer Tal 6, 76761 Rülzheim
loubna.ghammam@itk-engineering.de
[3] Université de Toulon, Avenue de l'Université BP20132 83957 LA GARDE CEDEX - FRANCE.
nicola.meloni@univ-tln.fr
[4] P.o.Box 39 Bambili, Higher Teacher Training College, Bambili The University of Bamenda.
fouotsa.emmanuel@uniba.cm

Abstract. In this paper, we propose two new versions of Miller algorithm in order to secure pairing computations against existing side-channel attacks (SCA). We have chosen to use the co-Z arithmetic on elliptic curves from which we derive two methods for pairing computations: one based on Euclidean addition chains and one based on Zeckendorf representation. We show that our propositions are resistant to existing side-channel attacks against pairing-based cryptography. We consider differential power analysis and fault attacks. The complexities of our solutions are compared with state-of-the-art one. We demonstrate that our new proposed versions are more efficient by **17%**.
Keywords: Miller algorithm, Euclidean addition chains, Fibonacci number, optimal Ate pairing, side-channel attacks.

1 Introduction

Pairing-based cryptography (PBC) provides several protocols such as short signature protocols and hierarchical encryption [1,2], making it a promising tool for the Internet of things (IoT) or cloud computing. In the first case, pairings would be implemented on constrained devices [3] such as microcontrollers and would be subject to invasive and non-invasive Side-Channel Attacks (SCA) [4,5,6,7,8,9]. Efficient and secure pairing computation has thus been an active research area [10,8,11].

The most efficient pairing computation algorithms are usually based on the Tate model [12] and rely on the Miller algorithm in order to compute the rational function $f_{s,Q}$ such that $\mathrm{Div}(f_{s,Q}) = s(Q)-([s]Q)-(s-1)(\mathcal{O})$, where P and Q are two points of an elliptic curve defined over a finite field $\mathbb{F}_q$ and $\mathcal{O}$ represents the identity element for the group of rational points of the elliptic curve. This function is computed thanks to the *double-and-add* loop called Miller loop [13] and followed by a final exponentiation.

Such algorithms are known to be natural targets to SCA ([14,15,16,17,18,9]). Our contribution is to provide a new approach to Miller loop to replace the usual *double-and-add* loop by a loop based on addition chains. Such approaches, e.g., based on the *double-and-add* algorithm, have been proven to be an interesting alternative to standard scalar multiplication methods when security is in balance with speed, especially when combining to co-Z arithmetic [19]. Concretely, we provide two versions of the Miller algorithm, one based on Euclidean addition chain and the other version based on the Zeckendorf representation of the parameter controlling the number of iterations. We show that both versions are resistant to Side-Chanel Attacks. Though the computational cost of these versions of the Miller algorithm is higher than that of the classic Miller algorithm, they are the most efficient among other countermeasures like masking against SCA.

The paper is organized as follows. Section 2 and 3 are devoted to the necessary mathematical background on pairing computations, co-Z arithmetic, and scalar multiplication using Euclidean additions chains (EAC). In Section 4, we present the new versions of the Miller algorithm, their complexities, and we compare them with the standard version. Section 6 is devoted to the state of the art on SCA and in Section 7 we analyze the security of our algorithms against some of those attacks. Finally, in Section 8, we provide detailed comparisons between our counter-measure, and other counter-measures used in the literature.

Notations:
In this paper we denote by:

- M_e a multiplication in the field $\mathbb{F}_{p^e}$.
- S_e a squaring in the field $\mathbb{F}_{p^e}$.

A multiplication, a squaring and an inversion in $\mathbb{F}_p$ are respectively denoted by M, S and I.

2 Background on pairing computations

Let E be an elliptic curve defined over a prime field $\mathbb{F}_p$, with p a large prime number. Let r be a large prime divisor of $\#E(\mathbb{F}_p)$. In practice, (E, p, r) are provided using parametric families [20]. Let k be the smallest integer such that r divides $p^k - 1$, k is called the embedding degree of E relative to r. Let $G_1 = E(\mathbb{F}_p)[r]$ be the r-torsion subgroup of $E(\mathbb{F}_p)$. Let $G_2 = E'(\mathbb{F}_{p^{k/d}})[r] \cap \mathrm{Ker}(\pi_p - [p])$ where E' is the twist of E (if it exists) of degree d, π_p represents the Frobenius map over E and $[p]$ is the scalar multiplication by p over E. The subgroup of $\mathbb{F}^{\star}_{p^k}$ consisting of r-th roots of unity is denoted by $G_3 = \mu_r$. Let s be an integer constructed for the optimal Ate pairing [21] and depending on r and p. Then the (optimal) Ate pairing [21] is given by

$$T_r : G_1 \times G_2 \to G_3; (P, Q) \mapsto f_{s,Q}(P)^{(p^k-1)/r}.$$

The pairing computation is divided into two main steps [8, Chap. 3]. First, one has to compute $f_{s,Q}(P)$ using an iterative algorithm denoted as the Miller algorithm and described in Algorithm 1 below. As any algorithm based on the double-and-add algorithm, the overall cost directly depends on the length and Hamming weight of integer s.

Algorithm 1 : Miller algorithm $(P, Q) \to f_{s,Q}(P)$

Require: $P \in G_1, Q \in G_2$, $s = (s_{n-1}, \ldots, \ldots s_0)$: binary representation of s
Ensure: $f_{s,Q}(P) \in G_3 (\subset \mathbb{F}^{*}_{p^k})$
1: $f_1 \leftarrow 1$
2: $T \leftarrow Q$
3: **for** $i = n - 2$ **down to** 0 **do**
4: $T \leftarrow [2]T$
5: $f_1 \leftarrow f_1^2 \cdot l_1(P)$ where l_1 is the tangent to the point T.
6: **if** $r_i = 1$ **then**
7: $T \leftarrow T + Q$
8: $f_1 \leftarrow f_1 \cdot l_2(P)$ where l_2 is the line passing through T and P.
9: **end if**
10: **end for**
11: **return** f_1

The second step of computing the Tate pairing and its variants is the final exponentiation and consists of raising the final result of the main loop, $f_{s,Q}(P)$, to the power of $\frac{p^k-1}{r}$. The computation of this part can be simplified thanks to the k-th cyclotomic polynomial [3].

3 Co-Z arithmetic on elliptic curves

Given an elliptic curve E over a finite field $\mathbb{F}_q$ there are many ways to compute the addition of two points. To avoid the computation of inversions in $\mathbb{F}_q$ when considering affine coordinates, it is standard to represent the points of E in Jacobian coordinates. A point (x, y) given in affine coordinates is represented in Jacobian coordinates by a triplet $(X : Y : Z)$ such that $x = X/Z^2$ and $y = Y/Z^3$, the point at infinity being then represented by $(0 : 1 : 0)$. A typical point addition costs 11 multiplications (M) and 5 squarings (S) in $\mathbb{F}_q$. Considerable literature now exists on the various point addition formulae, coordinate systems and curve shapes that can be used in a different context [22].

In the particular case of two points sharing the same Z-coordinate, the addition can be performed using only 5M+2S [19]. This operation is usually referred to as co-Z addition or ZADD. Finding two points sharing the same Z-coordinate is very unlikely but, when combined with the right scheme, one can perform a whole scalar multiplication using the co-Z addition. In this work, we considered two of those schemes: one based on Euclidean addition chains (EAC) and another based on the Zeckendorf representation of integers [19].

3.1 Euclidean addition chains

A Euclidean addition chain (EAC) computing an integer k is a sequence $n = (n_1, n_2, \ldots, n_w)$ such that $n_1 = 1$, $n_2 = 2$, $n_3 = 3$, $n_w = k$ and $\forall\ 3 \leqslant i \leqslant w-1$, if $n_i = n_{i-1} + n_j$, for some $j < i-1$, then we have $n_{i+1} = n_i + n_{i-1}$ (big step) or $n_{i+1} = n_i + n_j$ (small step). Such an addition chain, computing an integer greater than 3, can easily be represented by a binary chain $c_4, c_5, \ldots, c_w$ representing the succession of big steps (noted 0) and

small steps (noted 1) starting from $(1, 2, 3)$. For instance, the EAC $(1, 2, 3, 4, 7, 10, 13, 23)$ can be represented as $(1, 0, 1, 1, 0)$. Another example is the Fibonacci sequence $(1, 2, 3, 5, 8, 13, 21, \dots)$ which is only made of big steps.

Example 31 *The addition chain* (1, 2, 3, 4, 7, 11, 15, 19, 34, 53) *is an Euclidean addition chain computing* 53*. For example, in step 4, we have computed* $4 = 3 + 1$*. Then for step 5, we must add 3 or 1 to 4, that means that from step 4, we can compute* $7 = 4 + 3$ *or* $5 = 4 + 1$*. In our example, we have chosen to compute,* $7 = 4 + 3$ *(which is a big step). For step 6, we have computed* $11 = 7 + 4$ *(big step), etc.*

Finding such chains is simple: choose an integer g co-prime with k and then apply the subtractive form of Euclid's algorithm to those numbers.

Example 32 *We have presented* 53 *as an Euclidean addition chain. Let* $g = 34$*,* $\gcd(34,\ 53) = 1$*. Let apply now the subtractive form of Euclid's algorithm:*

$$\begin{aligned} 53 - 34 &= 19 \\ 34 - 19 &= 15 \\ 19 - 15 &= 4 \\ 15 - 4 &= 11 \\ 11 - 4 &= 7 \\ 7 - 4 &= 3 \\ 4 - 3 &= 1 \\ 3 - 1 &= 2 \\ 2 - 1 &= 1 \\ 1 - 1 &= 0 \end{aligned}$$

The Euclidean addition chain computing 53 *is* (1, 2, 3, 4, 7, 11, 15, 19, 34, 53)*. We find this chain by reading the first number of each line of the subtractive form of Euclid's algorithm.*

It has been shown that those chains can be an efficient alternative to the classical binary chain used in most scalar multiplication algorithms when side-channel attack resistance is an issue [19,23]. Algorithm 2 [19] describes how to perform a scalar multiplication using an EAC.

The main drawback of this approach is that, although it is easy to find an EAC computing a given integer k, finding a short one seems to be a challenging problem [24] making it difficult to use EAC for scalar multiplication directly. However, in the case of PBC, the Miller algorithm uses a fixed scalar which means that it is worth investigating finding an efficient EAC computing the parameter s. However, this method will become less efficient when s grows in size as finding a short EAC will become harder.

Algorithm 2 [19] Calculation of $[k]P$ where k is presented as an Euclidean Additions Chain.

Require: $P \in E, c = (c_4, \dots, \dots c_w)$ an EAC computing k
Ensure: $[k]P \in E$
  $(U_1, U_2) \leftarrow ([2]P, P)$
  for $i = 4$ **to** w **do**
    if $c_i = 0$ **then**
      $(U_1, U_2) \leftarrow (U_1 + U_2, U_1)$
    else
      $(U_1, U_2) \leftarrow (U_1 + U_2, U_2)$
    end if
  end for
  $U_1 \leftarrow U_1 + U_2$
  return U_1

3.2 Zeckendorf Representation

Let k be an integer and $(F_i)_{i \geq 0}$ the Fibonacci sequence, a classical result states that k can be uniquely written in the form $k = \sum_{i=2}^{l} d_i F_i$, with $d_i \in \{0, 1\}$ and $d_i d_{i+1} = 0$ [25]. An integer k written in this form is said to

be in Zeckendorf representation and will be denoted as $k = (d_{l-1}, \ldots, d_2)_{\mathcal{Z}}$. Such a representation is easy to compute as it can be obtained using a greedy algorithm [19].

As mentioned in [25] and [19], the disadvantage of the Zeckendorf representation is that it requires 44% more digits than the binary method requires. However, contrary to EAC, it is easy to bound its length [25]. Given the Zeckendorf representation of k, one can perform a scalar multiplication using the following Algorithm 3 [19].

Algorithm 3 [19] Computing $[k]P$ using Zeckendorf representation

Require: $P \in E, k = (d_n, \ldots, \ldots d_2)_Z$ Zeckendorf representation of k with $d_n = 1$
Ensure: $[k]P \in E$
1: $(U_1, U_2) \leftarrow (P, P)$
2: **for** $i = n - 1$ **to** 2 **do**
3: **if** $d_i = 1$ **then**
4: $(U_1, U_2) \leftarrow (U_1 + P, U_2)$
5: **end if**
6: $(U_1, U_2) \leftarrow (U_1 + U_2, U_1)$
7: **end for**
8: **return** U_1

4 Co-Z approach to the Miller Algorithm

In this section, we will present two variants of the Miller algorithm. We should remember that the basic idea of Miller's algorithm is to consider the binary representation of the Miller loop length parameter s and apply the *double-and-add* algorithm. The main idea of our work is to consider different representations of s to take advantage of co-Z formulae adapted for pairing computations.

4.1 Miller-Euclide

The idea is to adapt the scalar multiplication scheme described in Algorithm 2 to Miller loop and take advantage of the co-Z coordinates to limit the computational cost. Our approach, referred to as Miller-Euclide, is shown in Algorithm 4.

Algorithm 4 : Miller-Euclide, The computation of $f_{s,Q}(P)$ using an Euclidean addition chain.

Require: $P \in G_1$, $Q \in G_2$, $c = (c_4, \ldots, \ldots c_w)$ the euclidean addition chain computing s,
Ensure: $f_{s,Q}(P)$
1: $(T_1, T_2) \leftarrow ([2]Q, Q)$
2: $(f_1, f_2) \leftarrow (l_{Q,Q}(P), 1)$
3: **for** $i = 4$ **to** w **do**
4: **if** $c_i = 0$ **then**
5: $(f_1, f_2) \leftarrow (f_1 \times f_2 \times \ell_{T_1,T_2}(P), f_1)$
6: $(T_1, T_2) \leftarrow (T_1 + T_2, T_1)$
7: **else**
8: $(f_1, f_2) \leftarrow (f_1 \times f_2 \times \ell_{T_1,T_2}(P), f_2)$
9: $(T_1, T_2) \leftarrow (T_1 + T_2, T_2)$
10: **end if**
11: **end for**
12: $f_1 \leftarrow f_1 \times f_2 \times \ell_{T_1+T_2,T_1}(P)$
13: **return** f_1

First, let us notice that the initialization consists of one doubling and the evaluation of the tangent line passing through Q. It has been shown [19] that the doubling is compatible with the co-Z coordinates without additional cost so that the initialization can be performed in 2 multiplications and 5 squarings for the doubling. We add to this cost 3 multiplications and one squaring for the evaluation of the tangent. The details of this computation are presented in Algorithm 5.

The main loop of the new Miller-Euclide algorithm is only made of addition steps, one for each bit of the EAC, so that the computational cost of our method is simply linked to the length of the EAC representing s.

Algorithm 5 Initialization step of Algorithm 4 lines 1 and 2

Require: $P = (x_P, y_P)$ and $Q = (X_Q, Y_Q, Z_Q)$
Ensure: $T_1 = [2]Q, T_2 = Q$ sharing the same z-coordinate Z', $f_1 = \ell_{Q,Q}(P)$, $X_P = x_P Z'^2$ and $Y_P = y_P Z'^3$

1: $A \leftarrow X_Q^2$ $(S_{k/d})$
2: $B \leftarrow Y_Q^2$ $(S_{k/d})$
3: $C \leftarrow B^2$ $(S_{k/d})$
4: $D \leftarrow 2((X_1 + B)^2 - A - C)$ $(S_{k/d})$
5: $E \leftarrow 3A$
6: $F \leftarrow E^2$ $(S_{k/d})$
7: $X_{2Q} \leftarrow F - 2D$
8: $Y_{2Q} \leftarrow E(D - X_{2Q}) - 8C$ $(M_{k/d})$
9: $Z_{2Q} \leftarrow 2Y_Q Z_Q$ $(M_{k/d})$
10: $X_P \leftarrow x_P Z_{2Q}^2$ $(S_{k/d} + M_{k/d})$
11: $Y_P \leftarrow y_P Z_{2Q}^3$ $(2M_{k/d})$
12: $f_1, f_2 \leftarrow Y_p + X_p - 3A^2 + 16C, 1$
13: $T2 = (X_2, Y_2) \leftarrow D, 8C$
14: $T1 = (X_1, Y_1) \leftarrow X_{2Q}, Y_{2Q}$

In order to evaluate the cost of our method we must evaluate the cost of a full addition step. Let $T_1 = (X_1, Y_1, Z)$ and $T_2 = (X_2, Y_2, Z)$ be two points with the same z-coordinate. From [19] we have that the coordinates (X_3, Y_3, Z_3) of the point $T_3 = T_1 + T_2$ can be computed using the following equations:

$$\begin{aligned}
&A = (X_2 - X_1)^2,\ B = X_1 A,\ C = X_2 A,\ D = (Y_2 - Y_1)^2 \\
&X_3 = D - B - C\ , \\
&Y_3 = (Y_2 - Y_1)(B - X_3) - Y_1(C - B)\ , \\
&Z_3 = Z(X_2 - X_1)\ .
\end{aligned}$$

This computation requires 5 multiplications and 2 squarings over $\mathbb{F}_{p^{k/d}}$. On top of that the quantities $X_1 A = X_1(X_2 - X_1)^2$ and $Y_1(C - B) = Y_1(X_2 - X_1)^3$ computed during the addition can be seen as the x and y-coordinates of the point $(X_1(X_2 - X_1)^2, Y_1(X_2 - X_1)^3, Z(X_2 - X_1)) \sim (X_1, Y_1, Z)$. Thus it is possible to add P_1 and $P_1 + P_2$ with the same formulae during the next iteration of the main loop.

Let us now consider the value of $l_{T_1,T_2}(P)$ given by the equation of the line joining T_1 and T_2 evaluated on $P = (x_P, y_P)$. The standard equation of such line in affine coordinates is given by

$$y_p = \frac{y_1 - y_2}{x_1 - x_2} x_P + \frac{y_2 x_1 - y_1 x_2}{x_1 - x_2}, \tag{1}$$

To convert this equation to co-Z coordinate, we just need to apply the transformation $(x_i, y_i) \mapsto \left(\frac{X_i}{Z^2}, \frac{Y_i}{Z^3}\right)$ which gives us

$$\begin{aligned}
y_p &= \frac{Y_1 - Y_2}{Z(X_1 - X_2)} x_P + \frac{Y_2 X_1 - Y_1 X_2}{Z^3(X_1 - X_2)} \\
&\Leftrightarrow y_p(X_1 - X_2)Z^3 = (Y_1 - Y_2)Z^2 x_p + Y_2 X_1 - Y_1 X_2 \\
&\Leftrightarrow y_p(X_1 - X_2)Z^3 = (Y_1 - Y_2)(x_p Z^2 - X_1) - Y_1(X_1 - X_2) \\
&\Leftrightarrow y_p Z'^3 = (Y_1 - Y_2)(x_p Z'^2 - X_1') - Y_1'
\end{aligned}$$

where $X_1' = X_1(X_1 - X_2)^2$, $Y_1' = Y_1(X_1 - X_2)^3$ and $Z' = Z(X_1 - X_2)$. Now we remark that all those values have been computed during the point addition as well as the values of $(X_1 - X_2)^2$ and $(X_1 - X_2)^3$. If we suppose that the values of $y_p Z^3$ and $x_p Z^2$ were stored from the previous iteration of the main loop, the evaluation of the line joining the points T_1 and T_2 required exactly 3 multiplications over $\mathbb{F}_{p^{k/d}}$. Moreover, the value of $Z' = Z(X_1 - X_2)$ itself is not needed during the whole process, so that we can spare one multiplication per iteration, for a total cost of 7 multiplications and 2 squarings over $\mathbb{F}_{p^{k/d}}$.

Finally, we have to compute $f_1 \times f_2 \times \ell_{T_1,T_2}(P)$ which requires one sparse multiplication between elements of $\mathbb{F}_{p^{k/d}}$ and $\mathbb{F}_{p^k}$ and one full multiplication over $\mathbb{F}_{p^k}$.

Algorithm 6 sums up the computations required to perform one addition step with their respective costs.

Therefore, each addition step of Algorithm 4 requires $M_k + M_{k,k/d} + 7M_{k/d} + 2S_{k/d}$ operations. At the end the total cost of the algorithm is $5M_{k/d} + 6S_{k/d} + (n-2)(M_k + M_{k,k/d} + 7M_{k/d} + 2S_{k/d})$.

Algorithm 6 Computing one addition step of Algorithm 4

Require: T_1, T_2 with same z-coordinate Z, $X_p = x_p Z^2, Y_p = y_P Z^3$ and $f_1, f_2 \in \mathbb{F}_{p^k}$
Ensure: $Z' = Z(X_1 - X_2), T_3 = T_1 + T_2, T_1 = (X_3, Y_3, Z'), T_2 = (X_1 Z'^2 Y_1 Z'^3, Z'), X_p = x_p Z'^2, Y_p = y_P Z'^3, f_1 = f_1 \times f_2 \times \ell_{T_1,T_2}(P)$

1:	$A \leftarrow (X_1 - X_2)^2$	$(S_{k/d})$
2:	$B \leftarrow X_1 A$	$(M_{k/d})$
3:	$C \leftarrow X_2 A$	$(M_{k/d})$
4:	$D \leftarrow (Y_1 - Y_2)^2$	$(S_{k/d})$
5:	$E \leftarrow C - B$	
6:	$F \leftarrow Y_1 E$	
7:	$X_3 \leftarrow D - B - C$	
8:	$Y_3 \leftarrow (Y_2 - Y_1)(B - X_3) - F$	$(2M_{k/d})$
9:	$X_P \leftarrow X_P A$	$(M_{k/d})$
10:	$Y_P \leftarrow Y_P E$	$(M_{k/d})$
11:	$L \leftarrow Y_P - (Y_1 - Y_2)(X_P - X_3) - Y_3$	$(M_{k/d})$
12:	$f_3 \leftarrow f_1 L$	$(M_{k,k/d})$
13:	$f_3 \leftarrow f_1 f_2$	(M_k)
14:	$f_1, f_2 \leftarrow f_3, f_1$	
15:	$X_2, Y_2 \leftarrow B, F$	
16:	$X_1, Y_1 \leftarrow X_3, Y_3$	

4.2 Miller-Fibonacci

Let us now adapt Algorithm 3 to the Miller loop to present our new Algorithm 7 which computes $f_{s,Q}(P)$ using the Zeckendorf representation of s.

Algorithm 7 : Miller-Fibonacci: The computation of $f_{s,Q}(P)$ using Fibonacci sequences

Require: $P \in G_1$, $Q \in G_2$, $(d_n, \ldots, \ldots d_2)$ The Zeckendorf representation of s,
Ensure: $f_{u,Q}(P)$

```
(T1, T2) ← (Q, Q)
(f1, f2) ← (1, 1)
for i = n − 1 down to 2 do
    if d_i = 1 then
        (f1, f2) ← (f1 × l_{T1,T2}(P), f2)
        (T1, T2) ← (T1 + Q, T2)
    end if
    (f1, f2) ← (f1 × f2 × l_{T1,T2}(P), f1)
    (T1, T2) ← (T1 + T2, T1)
end for
return f1
```

First, let us note that the first computation will be a doubling instead of an addition, independently of the value of s_{n-1}, but for the sake of simplicity, we only use the additive notation. For this particular step, we use Algorithm 5.

Each step of this algorithm is similar to Miller-Euclide. If the current digit is a 0, we use the procedure described in Algorithm 6. If the current digit is a 1 then we have to perform an additional sparse multiplication and then the evaluation of the line passing through two points of the elliptic curve. To do so, we must first upgrade the coordinates of T_1 and Q so that they share the same z-coordinate as well as the values of X_p and Y_p (this is easily done in 7 multiplications and 2 squarings) and then use the formulae given in Algorithm 6. On top of that, we have to keep track of the z-coordinate of T_1 during the whole process which adds another multiplication. In the end the cost when the $d_i = 0$ is $M_k + M_{k,k/d} + 8M_{k/d} + 2S_{k/d}$ plus $7M_{k/d} + 2S_{k/d}$ if $d_i = 1$. The total cost of Algorithm 7 is then $5M_{k/d} + 6S_{k/d} + (n-2)(M_k + M_{k,k/d} + 7M_{k/d} + 2S_{k/d}) + hz(s)(7M_{k/d} + 2S_{k/d})$ where $hz(s)$ is the hamming weight of the Zeckendorf representation of s.

5 Comparison

In this section, we compare three versions of the Miller algorithm that allow us to compute the rational function $f_{s,Q}$ evaluated at P: Miller's algorithm, Miller-Euclide, and Miller-Fibonacci. For the sake of simplicity, we

restraint our comparisons to the computation of the Optimal Ate pairing over BN curves [26] and BLS 12 curves [27].

After Barbulescu and Duquesne results presented in [28], it is recommended to use BLS12 for computing Optimal Ate pairing for the 128-bit security level instead of BN curves. However, BN curves are still considered in practice for several schemes [29,30]. That's why, in this paper, we consider the two curves in our computation.

Both of these curves have the embedding degree $k = 12$, therefore, the arithmetic in their extension tower has the same complexity (since we don't consider additions in our cost evaluation). The tower extension used for pairing computation on both curves is given by: The field $\mathbb{F}_{p^{12}}$ is built using the following extension tower.

- $\mathbb{F}_{p^2} = \mathbb{F}_p[\alpha]/(\alpha^2 + 1)$
- $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[\beta]/\left(\beta^3 - (\alpha + 1))\right)$
- $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[\gamma]/\left(\gamma^2 - \beta\right)$

In Table 1, we also present the cost of each operation needed for computing Miller algorithm using projective coordinates [31] and also the cost of the operations needed for our new versions of Miller's algorithm.

Operation	Cost in $\mathbb{F}_p$
Multiplication in $\mathbb{F}_{p^{12}}$	$54\,M$
Sparse Multiplication in $\mathbb{F}_{p^{12}}$	$39\,M$
Squaring in $\mathbb{F}_{p^{12}}$	$36\,M$
Multiplication in $\mathbb{F}_{p^2}$	$3\,M$
Squaring in $\mathbb{F}_{p^2}$	$2\,M$
Addition step for Miller classic (Projective coordinates)	$80\,M$
Doubling step for Miller classic (Projective coordinates)	$100\,M$
Initialization step for Miller-Euclide/Fibonacci	$27\,M$
Addition step for Miller-Euclide	$118\,M$
Addition step for Miller-Fibonacci(for $u_i = 0$)	$121\,M$
Addition step for Miller-Fibonacci (for $u_i = 1$)	$146\,M$

Table 1. Cost of necessary operations for Miller loop

Note that these two curves, BN and BLS12, have a twist of degree $d = 6$ and the twist isomorphism is given by the following map.

$$\begin{aligned} \Psi : E'\left(\mathbb{F}_{p^2}\right) &\to E(\mathbb{F}_{p^{12}}) \\ (x_{Q'}, y_{Q'}) &\mapsto (\gamma^2 x_{Q'}, \gamma^3 y_{Q'}) \end{aligned}$$

5.1 BN curves

A Barreto-Naehrig (BN) curve [26] is an elliptic curve $E : y^2 = x^3 + b$ defined over a finite prime field $\mathbb{F}_p$ such that

- $b \in \mathbb{F}_p$,
- $E(\mathbb{F}_p)$ has prime order $n = \#E(\mathbb{F}_p)$,
- $p = p(u) = 36u^4 + 36u^3 + 24u^2 + 6u + 1$, and
  $n = n(u) = 36u^4 + 36u^3 + 18u^2 + 6u + 1$ for some $u \in \mathbb{Z}$.

We choose the following elliptic curve which is efficient in practice [31]:

$$E(\mathbb{F}) : \; y^2 = x^3 + 2.$$

The twisted elliptic curve is defined by the equation:

$$E'(\mathbb{F}_{p^2}) : y^2 = x^3 + (1 - i).$$

Recall that the Optimal Ate pairing over BN curves is the following map:

$$\begin{aligned} E(\mathbb{F}_p)[r] \times \Psi(E'(\mathbb{F}_{p^2}))[r] &\longrightarrow \mathbb{F}^{\star}_{p^{12}} \\ (P, Q) &\longmapsto = \left((f_{6u+2,Q}(P) l_{[6u+2]Q,\pi_p(Q)}(P) l_{[6u+2]Q,\pi_p^2(Q)}(P))\right)^{\frac{p^{12}-1}{r}} \end{aligned}$$

Using the costs presented in Table 1, we obtain the comparison between the different versions of Miller's algorithm to compute $f_{6u+2,Q}(P)$, in terms of operations in $\mathbb{F}_p$. This comparison is presented in Table 2. We recall that the pairing parameter in our case is $s = 6u + 2$ with $u = -2^{114} + 2^{101} - 2^{14} - 1$. In this case, it is possible to find a EAC of length 168 (using the subtractive euclidean algorithm with parameter s with $s = 17460112080228648700297903040853251 9$ when we consider $u = 4708509132198764084405361370957937 2$). Moreover, the Zeckendorf representation of $s = 6u + 2 = Fibo(169) + Fibo(61) + Fibo(99)$ with a chosen parameter $u = 155337012968972389620712357587721 59$, has a length of 168 and its hamming weight is 3. Thus, we have:

Method	Complexity in $\mathbb{F}_p$	
Miller classic Algo. 1	$12068\,M$	
Miller-Euclide Algo. 4	$19991\,M$	$+\mathbf{65,5}\%$
Miller-Fibonacci Algo. 7	$20683\,M$	$+\mathbf{71,3}\%$

Table 2. Cost of each Miller algorithm for BN curves

From this Table 2, we can notice that the classical Miller algorithm is more efficient than our two new versions. However, it is not secure when considering DPA attack. We show in the next section that our method protects against this attack.

5.2 BLS 12 Curves

In 2002, Barreto, Lynn and Scott presented in [27] a method to generate pairing-friendly elliptic curves over a prime field $\mathbb{F}_p$ with embedding degree $k = 12$. BLS12 are defined over $\mathbb{F}_p$ by the following equation:

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

and by a parameter $u \in \mathbb{Z}$ such that:

$$\begin{cases} p = (u-1)^2(u^4 - u^2 + 1)/3 + u \\ r = u^4 - u^2 + 1 \\ t = u + 1 \end{cases} \tag{2}$$

where t is the trace of the Frobenius map on the curve. The parameter u is chosen such that p and r are prime and have the sizes corresponding to the desired security level. For the 128-bit security level and as recommended in [28], p and r are of at least 461 and 308 and the proposed parameter $u = -2^{77} + 2^{50} + 2^{33}$. The choosed BLS12 elliptic curve is defined over $\mathbb{F}_{\text{ı}}$ by $E(\mathbb{F}) :\ y^2 = x^3 + 4$ which admits a twist of degree $d = 6$ and given by $E'(\mathbb{F}_{p^2}) : y^2 = x^3 + 4(1 - i)$.

The Optimal Ate pairing on BLS12 curves is defined by the following map.

$$\begin{aligned} E(\mathbb{F}_p)[r] \times \Psi(E'(\mathbb{F}_{p^2}))[r] &\longrightarrow \mathbb{F}^{\star}_{p^{12}} \\ (P, Q) &\longmapsto = ((f_{u,Q}(P))^{\frac{p^{12}-1}{r}}. \end{aligned}$$

Using the costs presented in Table 1, we obtain the comparison between the different versions of the Miller algorithm to compute $f_{u,Q}(P)$, in terms of operations in $\mathbb{F}_p$. This comparison is presented in Table 3. We recall that the pairing parameter in our case is $u = -2^{77} + 2^{50} + 2^{30}$. In this case, it is possible to find a EAC of length 112 (using the subtractive euclidean algorithm with parameter $u' = 2040354807236363787925 33$ and $s' = 126100861998131272870737$). Note that u' is chosen such that p and r are primes. Moreover, the Zeckendorf representation of $u = 704925247670891258604 97 = Fibo(3) + Fibo(7) + Fibo(24) + Fibo(111)$ has length 111 and its hamming weight is 4 (u is chosen such that we obtain an efficient Zeckendorf representation where p and r are primes. Thus, we have:

From this Table 3, we can notice that the classical Miller algorithm is more efficient than our two new versions. However, it is not secure when considering DPA attack. We show in the next section that our method protects against this attack.

Method	Complexity in $\mathbb{F}_p$	
Miller classic Algo. 1	$7438\,M$	
Miller-Euclide Algo. 4	$13147\,M$	$+\mathbf{76,8}\%$
Miller-Fibonacci Algo. 7	$13936\,M$	$+\mathbf{87,3}\%$

Table 3. Cost of each Miller algorithm for BLS12

6 SCA against Miller's algorithm

The SCA re the perturbation analysis that are invasive and observation analysis that are non-invasive. Non-invasive attacks against pairing, such as DPA, CPA, template attack, have been widely studied [14,15,4,32,6,33,34,17,9], so as the invasive attacks (basically fault attacks) [7,8]. In this section, we briefly recall the non-invasive attacks against pairing, then, we will illustrate the fact that our new algorithms seem to be resistant to existing attacks.

6.1 Non-invasive attacks

Non-invasive attacks can use for instance the power consumption or electromagnetic emission of a device. Those attacks are also called observation analysis, they include Differential Power Analysis, Correlation Power Analysis, Template attacks and can be performed in vertical or horizontal mode. We will describe the resistance of our algorithms by using the DPA attack model to resume the non invasive attacks. Those attacks can allow an attacker to compute the intermediate values within cryptographic computations through statistical analysis of data collected from multiple cryptographic operations. In pairing-based cryptography, the secret is one of the inputs, either the point P either the point Q. The public data is the other point. The vulnerabilities of a pairing implemented with a classical Miller algorithm have been illustrated in several articles [35,36,37,4,32,33,34,38,39,17]. The scheme is as follow:

1. First choose the secret (P or Q).
2. Then find during the algorithm an operation that involves both the secret data and a data that can be manipulated freely during the attack. This operation is the target of the attack. In pairing-based cryptography, the freely chosen data are the coordinates of the public point.
3. For a vertical attack: the operation is executed at a specific time t_0, the statistical analysis is performed on a collection of traces for the same secret point and hundred to millions of public point.
4. For a horizontal attack: the same secret data is used more than once during a single algorithm execution, the statistical analysis is performed using one public entry at different times t_i.

6.2 Existing Countermeasure

To avoid SCA in pairing-based cryptography, several counter-measures have been proposed in the literature:

1. The counter-measure proposed by Öztürk et al. in [40]. The principle of this counter-measure is to avoid any perturbation against Miller's algorithm using resilient counters. The disadvantage of this counter-measure is that it is a proof of concept implementation and doesn't have a theoretical basis for measuring its security.
2. The second counter-measure is proposed by Gosh et al. in [41]. It consists of implementing a modified version of the Miller algorithm. Unfortunately, this method introduces an additional calculation which is expensive and which, moreover, does not really improve security for the Miller algorithm.
3. The use of the homogeneity property of projective coordinates (or Jacobian coordinates) of the point P or of the point Q, is a counter-measure against this attack on Miller's algorithm [35]. This counter-measure causes a modification of the result of the Miller algorithm: thus, we obtain a multiple of the final result of Miller algorithm. However, this is not a problem because this multiple being in the subfield of $\mathbb{F}_{p^k}$, will be eliminated by the final exponentiation.
   The advantage of this counter-measure is that it is not expensive. However, unfortunately, El Mrabet et al. have shown in [42] that this is not enough to protect Miller's algorithm.
4. Another efficient counter-measure is to choose two integers a et b such that $a \times b = 1 \mod r$ [35]. Then, the idea is to compute the pairings between the points $[a]P$ and $[b]Q$ instead of computing the pairing between P and Q. This operation is possible thanks to the bilinearity of pairings, we have:

$$e([a]P,\ [b]Q) = e(P,\ Q)^{ab}.$$

Also, thanks to the final exponentiation, the exponent, ab will be canceled. Finally, we obtain:

$$e(P,\, Q)^{ab\times(\frac{p^k-1}{r})} = e(P,\, Q)^{(\frac{p^k-1}{r})}.$$

5. Finally, we cite the most considered counter-measure: **masking** [35]. When the secret is the point P, this counter-measure consists of choosing a random point R of the elliptic curve E such that $e(P, R)$ is defined, then computing the pairing between P and Q passing through the point R. Using the property of bilinearity, we obtain:
$$e(P,\, Q) = e(P,\, Q + R) \times e(P,\, -R).$$
Therefore, this counter-measure consists of computing two pairings instead of one and then performing their product.

The last two counter-measures are the most considered in practice. In a theoretical study of the various existing counter-measures, El Mrabet et al. suggested in [7] to consider the counter-measure by masking to protect the Miller algorithm against SCA attacks. The counter-measure based on the bilinearity of pairings implies the computation of two scalar multiplications, one on $E(\mathbb{F}_p)$, and one on $E(\mathbb{F}_{p^k})$, both of them are scalar multiplication by an integer of order r. Make these two computations secure against side channel attack is a classical problem in elliptic curve cryptography. In [43], Feix at al. demonstrate that even a secure scalar multiplication can be sensitive to side channel attack. The use of that counter-measure in pairing-based cryptography would imply to add two other sensitive computations with their counter-measure [19,23,44]. On the other side the masking counter-measure, imply one addition over E and in the worst case two pairing computations shorter than the scalar multiplication. As a consequence, we consider the masking counter-measure in the sequel of the article.

7 Resistance of our methods against non-invasive attacks

Since Miller-Fibonacci algorithm is based on the operation used in the Miller-Euclide so that proving that it is secure against a non invasive would show that the Miller-Euclide is also secured.

The Miller-Fibonacci algorithm is based on writing the pairing parameter s as a sequence of Fibonacci numbers. In our new algorithm, we have only additions steps. Therefore we only have to evaluate the line passing through the points T_1 and T_2 (or T_1 and Q) of respective coordinates $(X_1,\, Y_1,\, Z)$ and $(X_2,\, Y_2,\, Z)$ evaluated on the public point P.
Therefore, the only equation involving known data that the non invasive attack can use is the following one:

$$\begin{aligned} l_{T_1,T_2}(P) = {} & y_P Z^3 (X_1 - X_2)^3 - Y_1 (X_1 - X_2)^3 \\ & -(Y_1 - Y_2)\left(x_P Z^2 (X_1 - X_2)^2 - X_1 (X_1 - X_2)^2\right) \end{aligned}$$

The DPA attack model in pairing-based cryptography [35,4], is efficient when we have an operation that involved a known data and one of the temporary variables X_1, X_2, Y_1, Y_2 or Z. If the secret is the point Q, the known values are x_p and y_p. With a DPA we can find $Z^2(X_1 - X_2)^2$ using the multiplication $x_P Z^2 (X_1 - X_2)^2$ and $Z^3(X_1 - X_2)^3$ using $y_P Z^3 (X_1 - X_2)^3$. This gives us access to the value $Z(X_1 - X_2)$. Recall that our algorithm is described using the co-Z arithmetic, as a consequence the value of Z is the same for the points at each execution step of the Miller-Fibonacci algorithm. We analyze now which information can be found given two hypotheses on Z, either $Z_Q = 1$ or $Z_Q \neq 1$. If $Z_Q = 1$, i.e. $Z = 1$, then using a vertical DPA attack we can find the values $(X_1 - X_2)$, $(X_1 - X_2)^2$, and $(X_1 - X_2)^3$. Then using another DPA we can find either Y_1 using $Y_1(X_1 - X_2)^3$ or X_1 using $X_1(X_1 - X_2)^2$. With $Z = 1$ and X_1 or Y_1 and the elliptic curve equation, we can find Y_1 or X_1. The attack would be successful. But if $Z_Q \neq 1$, we cannot perform the attack. Indeed, we made the hypothesis that Q is secret, then Z_Q is also secret and different from 1. We do not know either Z, neither X_1 and X_2 as they are derived from Q. We can make hypotheses on the value of Z and then on $(X_1 - X_2)$, but we cannot try all the possible values of Z in $\mathbb{F}_p$. As a consequence, as long as $Z_Q \neq 1$, our Miller-Fibonacci algorithm, based on presenting the parameter pairing s with the Zeckendorf representation, is natively protected against a DPA attack. Indeed, once we eliminate the leakage of information during the operations $y_P Z^3 (X_1 - X_2)^3$ and $x_P Z^2 (X_1 - X_2)^2$, the other registers depend only on the secret point and consequently are protected against any vertical DPA analysis. In fact, our algorithm is also secure against horizontal attacks. Indeed, in horizontal attacks, the hypothesis is that a same entry is used more than once during one single execution of the algorithm. Using the co-Z arithmetic, we use the same Z coordinates during one step of the Miller algorithm. It is not sufficient to perform an horizontal SCA.

The same analysis can be done for Miller-Euclide which is also protected against SCA as long as $Z \neq 1$.

8 Comparison

In the Section 7, we have proved that our new algorithms resist against DPA attacks. In this section, we compare their respective complexities with a classical Miller algorithm protected again DPA attack. We choose to compare these two counter-measures in the context of computing Optimal Ate pairing on BN and BLS 12 curves for the reasons explained in the Introduction

Remark 81 *Our methods for computing the Miller loop using Euclidean addition chains or Fibonacci Sequences are available for computing all pairings and in any elliptic curve.*

We have to evaluate $f_{6u+2,Q}(P)$ using Miller algorithm where $6u+2$ is an integer of 114 bits and Hamming weight 4.
We present the following table the cost of our counter-measure by comparing it with the counter-measure by masking.

Algorithm	DPA secure	Cost BN	Cost BLS12
Miller	no	12068 M_p	7708 M_p
Miller+masking	yes	24136 M_p	15416 M_p
Miller-Euclide	yes	19991 M_p	13147 M_p
Miller-Fibonacci	yes	20683 M_p	13936 M_p

Table 4. Comparison between the costs of the counter-measures

Table 4 shows that our counter-measures are more efficient than the counter-measure proposed in the literature. The Miller-Euclide and the Miller-Fibonacci algorithm are, respectively, 16% and 17% faster than the Miller standard algorithm combined with the masking.

9 Conclusion

In this paper, we have proposed two new versions of Miller's algorithm: Miller-Euclide and Miller-Fibonacci. The complexity of these algorithms depends on the Miller loop parameter. Therefore, we showed how to choose a short representation for the parameter s which makes our algorithms as efficient in terms of complexity as possible.
We proved also in this paper that these two algorithms resist against Differential Power Analysis attacks.

We compare the complexity of our new algorithm Miller-Fibonacci with the most efficient counter-measure of the classical Miller algorithm. We proved that our proposal Miller-Fibonacci algorithm is more efficient than the counter-measure by masking about 10.7% in the worst case and in the best case about 20%. // We only investigate the vertical power analysis, we leave as an open problem the analysis of our algorithms against horizontal power analysis. Indeed, this kind of analysis needs implementations and practical attacks.

References

1. National Institute of Standards and Technology. http://csrc.nist.gov/publications/PubsSPs.html.
2. A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1984. Springer-Verlag New York, Inc.
3. Sylvain Duquesne and Loubna Ghammam. Memory-saving computation of the pairing final exponentiation on BN curves. *Groups Complexity Cryptology*, 8(1):75–90, 2016.
4. N. El Mrabet, G. Di Natale, and M.L. Flottes. A practical differential power analysis attack against the miller algorithm. In *PRIME 2009 - 5th Conference on Ph.D. Research in Microelectronics and Electronics, Circuits and Systems Magazine*, IEEE Xplore, 2009.
5. Johannes Blömer, Peter Günther, and Gennadij Liske. Improved side channel attacks on pairing based cryptography. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design*. Springer Berlin Heidelberg, 2013.
6. T. Unterluggauer and E. Wenger. Practical attack on bilinear pairings to disclose the secrets of embedded devices. In *2014 Ninth International Conference on Availability, Reliability and Security*, pages 69–77, 2014.
7. Nadia El Mrabet, Jacques J. A. Fournier, Louis Goubin, and Ronan Lashermes. A survey of fault attacks in pairing based cryptography. *Cryptography and Communications*, 7(1):185–205, 2015.

8. N. El Mrabet and M. Joye. *Guide to Pairing-Based Cryptography*. Chapman & Hall/CRC Cryptography and Network Security Series. CRC Press, 2017.
9. Damien Jauvart, Nadia El Mrabet, Jacques J. A. Fournier, and Louis Goubin. Improving side-channel attacks against pairing-based cryptography. *Journal of Cryptographic Engineering*, 2019.
10. Sanjit Chatterjee, Darrel Hankerson, and Alfred Menezes. *On the Efficiency and Security of Pairing-Based Protocols in the Ttype 1 and Ttype 4 Settings*, pages 114–134. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
11. R. Azarderakhsh, D. Fishbein, G. Grewal, S. Hu, D. Jao, P. Longa, and R. Verma. Fast software implementations of bilinear pairings. *IEEE Transactions on Dependable and Secure Computing*, 14(6):605–619, 2017.
12. Gerhard Frey, Michael Müller, and Hans-Georg Rück. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Trans. Information Theory*, 45(5):1717–1719, 1999.
13. Victor S. Miller. The weil pairing, and its efficient calculation. *J. Cryptology*, 17(4):235–261, 2004.
14. D. Page and F. Vercauteren. Fault and side channel attacks on pairing based cryptography. In *IEEE Transactions on Computers*, volume 55-9, pages 1075–1080, 2006.
15. Claire Whelan and Mike Scott. *Side Channel Analysis of Practical Pairing Implementations: Which Path Is More Secure?*, pages 99–114. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
16. T. Unterluggauer and E. Wenger. Practical attack on bilinear pairings to disclose the secrets of embedded devices. In *2014 Ninth International Conference on Availability, Reliability and Security*, pages 69–77, Sept 2014.
17. Damien Jauvart, Jacques J. A. Fournier, Nadia El Mrabet, and Louis Goubin. Improving side-channel attacks against pairing-based cryptography. In Frédéric Cuppens, Nora Cuppens, Jean-Louis Lanet, and Axel Legay, editors, *Risks and Security of Internet and Systems - 11th International Conference, CRiSIS 2016, Roscoff, France, September 5-7, 2016, Revised Selected Papers*, volume 10158 of *Lecture Notes in Computer Science*, pages 199–213. Springer, 2016.
18. Nadia El Mrabet, Louis Goubin, Sylvain Guilley, Jacques Fournier, Damien Jauvart, Martin Moreau, Pablo Rauzy, and Franck Rondepierre. *Physical Attacks*, chapter 12. CRC Press, 2016.
19. Nicolas Meloni. New point addition formulae for ECC applications. In *Arithmetic of Finite Fields, First International Workshop, WAIFI 2007, Madrid, Spain, June 21-22, 2007, Proceedings*, pages 189–201, 2007.
20. David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 23(2):224–280, 2010.
21. Frederik Vercauteren. Optimal pairings. *IEEE Trans. Information Theory*, 56(1):455–461, 2010.
22. Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2012.
23. Fabien Herbaut, Pierre-Yvan Liardet, Nicolas Meloni, Yannick Teglia, and Pascal Véron. Random euclidean addition chain generation and its application to point multiplication. In Guang Gong and Kishan Chand Gupta, editors, *Progress in Cryptology - INDOCRYPT 2010 - 11th International Conference on Cryptology in India, Hyderabad, India, December 12-15, 2010. Proceedings*, volume 6498 of *Lecture Notes in Computer Science*, pages 238–261. Springer, 2010.
24. D. Knuth and A. Yao. Analysis of the subtractive algorithm for greater common divisors. pages 4720–4722, December 1975.
25. E. Zeckendof, editor. *Représentations des nombres naturels par une somme de nombre de Fibonacci ou de nombres de Lucas*. Bulletin de la Société Royale des Sciences de Liège. 1972.
26. C. C. F. Pereira Geovandro, Marcos A. Simplício Jr., Michael Naehrig, and Paulo S. L. M. Barreto. A family of implementation-friendly BN elliptic curves. *Journal of Systems and Software*, 84(8):1319–1326, 2011.
27. Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In *Security in Communication Networks*, 2002.
28. Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. *Journal of Cryptology*, 2019.
29. Black-box wallets: Fast anonymous two-way payments for constrained devices. *IACR Cryptology ePrint Archive*, 2019:1199.
30. Sebastien Canard et Unida Diop et Nizar Kheir et Marie Paindavoine et Mohamed Sabt. Blindids: Market-compliant and privacy-friendly intrusion detection system over encrypted traffic. In *AsiaCCS 2017, Abu Dhabi, Emirats Arabes Unis, 2-6 avril, 2017*, pages 561–574.
31. Sylvain Duquesne, Nadia El Mrabet, Safia Haloui, and Franck Rondepierre. Choosing and generating parameters for low level pairing implementation on BN curves. *IACR Cryptology ePrint Archive*, 2015:1212, 2015.
32. Santosh Ghosh and Dipanwita Roychowdhury. Security of prime field pairing cryptoprocessor against differential power attack. pages 16–29. Springer, 2011.
33. Weibo Pan and WP Marnane. A correlation power analysis attack against Tate pairing on FPGA. *Reconfigurable Computing: Architectures, Tools and Applications*, pages 340–349, 2011.
34.
35. Dan Page and Frederik Vercauteren. Fault and Side-Channel Attacks on Pairing Based Cryptography. 2004.
36. Claire Whelan and Mike Scott. Side Channel Analysis of Practical Pairing Implementations: Which Path Is More Secure? *VIETCRYPT 2006*, pages 99–114, 2006.
37. Nadia El Mrabet. What about vulnerability to a fault attack of the miller's algorithm during an identity based protocol? In *ISA*, volume 5576 of *Lecture Notes in Computer Science*, pages 122–134. Springer, 2009.
38. Thomas Unterluggauer and Erich Wenger. Practical Attack on Bilinear Pairings to Disclose the Secrets of Embedded Devices. *ARES*, pages 69–77, 2014.

39. Nadia El Mrabet and Emmanuel Fouotsa. Failure of the point blinding countermeasure against fault attack in pairing-based cryptography. In Said El Hajji, Abderrahmane Nitaj, Claude Carlet, and El Mamoun Souidi, editors, *Codes, Cryptology, and Information Security*, pages 259–273, Cham, 2015. Springer International Publishing.
40. Erdinç Öztürk, Gunnar Gaubatz, and Berk Sunar. Tate pairing with strong fault resiliency. In *Fourth International Workshop on Fault Diagnosis and Tolerance in Cryptography, 2007, FDTC 2007: Vienna, Austria, 10 September 2007*, pages 103–111, 2007.
41. Santosh Ghosh, Debdeep Mukhopadhyay, and Dipanwita Roy Chowdhury. Fault attack, countermeasures on pairing based cryptography. *I. J. Network Security*, 12(1):21–28, 2011.
42. Ronan Lashermes, Marie Paindavoine, Nadia El Mrabet, Jacques J. A. Fournier, and Louis Goubin. Practical validation of several fault attacks against the miller algorithm. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2014, Busan, South Korea, September 23, 2014*, pages 115–122, 2014.
43. Benoit Feix, Mylène Roussellet, and Alexandre Venelli. Side-channel analysis on blinded regular scalar multiplications. In *Progress in Cryptology–INDOCRYPT 2014*, pages 3–20. Springer, 2014.
44. Apostolos P. Fournaris. *Fault and Power Analysis Attack Protection Techniques for Standardized Public Key Cryptosystems*, pages 93–105. Springer International Publishing, Cham, 2017.