Side-Channel Resistant Scalar Multiplication Algorithms over Finite Fields

Alexandre VENELLI (alexandre.venelli@atmel.com)*
François DASSANCE (francois.dassance@atmel.com)†

Abstract: Scalar multiplication is essential for any elliptic curve cryptosystems. Its efficient and secure implementation is even more important on embedded devices. We present a new scalar multiplication algorithm for elliptic curves over both types of finite fields, i.e. binary and prime fields. We combine the Meloni's simplified addition technique with a modified version of the Montgomery ladder multiplication algorithm. We benefit from the fast simplified addition and the side-channel resistant structure of Montgomery's multiplication. Initially on $\mathbb{F}_p$, we extend Meloni's technique on $\mathbb{F}_{2^m}$ in order to have a secure and efficient alternative to the powerful but patented Montgomery ladder on binary fields. We study the side-channel resistance of the Montgomery ladder, particularly against fault attacks, and we analyze how its resistance is applied our propositions. Finally, we compare our method with state-of-the-art algorithms at the same level of side-channel resistance.

Keywords: Elliptic curve, Scalar multiplication, Side-channel analysis

1 Introduction

Elliptic Curve Cryptosystems (ECC) are now a very attractive alternative to the classical public-key cryptography both in the cryptography research community and in the industry. The main reason for the attractiveness of ECC is that shorter key can be used for a similar level of security. This is particularly suitable for an implementation of embedded devices that have memory constraints. Because of the physical characteristics of these devices and their use in potentially hostile environments, they are particularly sensitive to side-channel attacks. Side-Channel Analysis (SCA) attacks use information observed during the execution of the algorithm to determine the secret key. There are three main categories of SCA attacks. Simple side-channel attacks, like Simple Power Analysis (SPA), analyze the trace of a single execution of the algorithm. Differential side-channel attacks, like Differential Power Analysis (DPA), compare the traces of multiple executions. Fault Analysis (FA) attacks take advantage of errors that occur while a cryptographic device is performing a private-key operation. Both Biel et al. [BMM00] and then Ciet et al. [CJ05] showed how to exploit errors in ECC. In general, SCA attacks represent a major threat to smart cards and mobile devices.

The most important operation of ECC is the scalar multiplication of an elliptic curve point P with a secret scalar factor k. This operation is often noted $[k]P$. Its compu-

* ATMEL Secure Microcontroller Solutions Zone Industrielle 13106 Rousset, FRANCE
IML - ERISCS Université de la Méditerranée Case 907, 163 Avenue de Luminy 13288 Marseille Cedex 09, FRANCE
† ATMEL Secure Microcontroller Solutions Zone Industrielle 13106 Rousset, FRANCE

tational cost is decisive in the overall efficiency of the ECC however implementing SCA countermeasures is very resource consuming. Numerous articles in the literature deal with securing the scalar multiplication against different SCA.

We are addressing the problem of finding a scalar multiplication algorithm that is both efficient and SCA resistant. We are particularly interested in the Montgomery ladder point multiplication as its structure is suitable for SPA and FA resistance. Another attractive feature of Montgomery's algorithm is that the y-coordinate of the elliptic point is not computed through the scalar multiplication. Hence, it can also be computationally efficient. Unfortunately, most of Montgomery's y-free methods are patented [VMAG99]. We provide alternative scalar multiplication algorithms that are SPA and FA resistant, like Montgomery's, while still efficient. We use Meloni's addition formula [Mel07] that is very efficient but requires the two input points to have the same Z-coordinate. Modifying the Montgomery ladder algorithm, we obtain a scalar multiplication algorithm that uses only Meloni-like additions both on $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$.

This article is organized as follows: we first briefly review elliptic curve arithmetic in Section 2. Then Section 3 presents classical side-channel resistant scalar multiplication algorithms on elliptic curves. In Section 4 we introduce our SCA resistant scalar multiplication algorithms and we compare its efficiency with other methods at the same level of side-channel resistance. Section 5 analyzes the security against side-channel attacks of our algorithms. Finally, Section 6 concludes the paper.

2 Elliptic curve arithmetic over finite fields

An elliptic curve E over a field K, denoted $E(K)$, is defined by the general Weierstrass equation:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{1}$$

where $a_1, a_2, a_3, a_4, a_6 \in K$ and $\Delta \neq 0$ with Δ the discriminant of E. The set of pairs (x, y) that solves (1) and the point at infinity ∞ form an abelian group $(E(K), +)$.

Let E be defined over a finite field $K = \mathbb{F}_p$, where p is a large prime, strictly greater than 3, that represents the number of elements of the field. In this case, the general Weierstrass equation simplifies to:

$$E : y^2 = x^3 + ax + b \tag{2}$$

where $a, b \in \mathbb{F}_p$ and $\Delta = 4a^3 + 27b^2 \neq 0 \bmod p$. There is similarly an abelian group structure $(E(\mathbb{F}_p), +)$.

The representation of points on an elliptic curve E with two coordinates (x, y), called affine coordinates, introduces field inversions in the computation of point addition and point doubling. Inversions over prime fields are often very expensive and are avoided as much as possible. It may be advantageous to represent points using projective coordinates of which several types have been proposed [BL07]. We consider here Jacobian projective coordinates because they offer a good compromise between computational costs and memory usage. A point P in Jacobian coordinates is noted $P = (X, Y, Z)$ and represents the affine point $(X/Z^2, Y/Z^3)$. Classical addition and doubling formulas can be found in [BL07]. We recall their complexity in terms of multiplications and squaring, respectively noted M and S, in the finite field $\mathbb{F}_p$.

- (General) Point addition in Jacobian coordinates: $11M + 5S \approx 16M$,
- Point doubling in Jacobian coordinates: $1M + 8S \approx 9M$.

Meloni [Mel07] introduced in 2007 a simplified point addition formula for generic elliptic curve over $\mathbb{F}_p$. Using Jacobian projective coordinates, a restriction is added on the input points of the addition. Let two points of an elliptic curve E be $P_1 = (X_1, Y_1, Z)$ and $P_2 = (X_2, Y_2, Z)$ with the same Z-coordinate, then the following point addition formula can be applied:

Simplified point addition on $\mathbb{F}_p$. Let $P_1 = (X_1, Y_1, Z)$, $P_2 = (X_2, Y_2, Z)$ both unequal to ∞ and $P_2 \neq \pm P_1$. Let $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$.

$$A = (X_2 - X_1)^2, \quad B = X_1 A, \quad C = X_2 A, \quad D = (Y_2 - Y_1)^2,$$

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

This special point addition only requires $5M + 2S \approx 7M$. It is even faster than the general point doubling in Jacobian coordinates. This formula is only useful if the two input points have the same Z-coordinates, which is very unlikely. However, Meloni noticed that, while computing the addition, one can easily modify the entry point P_1 so that P_1 and $P_1 + P_2$ have the same Z-coordinate at the end of the addition. We call this algorithm:

$$\texttt{SimpleAdd}(P_1, P_2) \rightarrow (\tilde{P_1}, P_1 + P_2).$$

Let us now consider an elliptic curve E over a finite field $\mathbb{F}_{2^m}$ given by the equation:

$$E : y^2 + xy = x^3 + ax^2 + b \tag{3}$$

where $a, b \in \mathbb{F}_2$. As previously, an abelian group structure can be defined $(E(\mathbb{F}_{2^m}), +)$. As in the $\mathbb{F}_p$ case, a large choice of representations of coordinates exists. We consider two types of projective coordinates systems because they offer both a good compromise between efficiency and memory usage. Let $P = (X, Y, Z)$ be a point in López-Dahab coordinates that represents the affine point $(X/Z, Y/Z^2)$ [BL07]. The complexities of the basic point operations are:

- (General) Point addition in López-Dahab coordinates: $13M + 4S \approx 17M$,
- Point doubling in López-Dahab coordinates: $3M + 5S \approx 8M$.

Although computationally interesting, we are not able to propose an efficient simplified addition similar to Meloni's using López-Dahab coordinates. We then consider projective Jacobian coordinates.

Let $P = (X, Y, Z)$ be a point in Jacobian coordinates that represents the affine point $(X/Z^2, Y/Z^3)$. The complexities are:

- (General) Point addition in Jacobian coordinates: $14M + 5S \approx 19M$,
- Point doubling in Jacobian coordinates: $4M + 5S \approx 9M$.

Let $P_1 = (X_1, Y_1, Z)$ and $P_2 = (X_2, Y_2, Z)$ be two points in Jacobian coordinates with the same Z-coordinate.

Simplified point addition on $\mathbb{F}_{2^m}$. Let $P_1 = (X_1, Y_1, Z)$, $P_2 = (X_2, Y_2, Z)$ both unequal to ∞ and $P_2 \neq \pm P_1$. Let $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$.

$$A = X_1 + X_2, \quad B = A^2, \quad C = AB, \quad D = Y_1 + Y_2, \quad E = CY_2, \quad F = BX_2,$$

$$G = FD, \quad H = Z_3 + D$$

$$\begin{cases} X_3 &= aZ_3^2 + DH + C, \\ Y_3 &= X_3 H + E + G, \\ Z_3 &= Z_1 A. \end{cases}$$

The simplified addition requires $7M + 2S \approx 9M$. As in $\mathbb{F}_p$, this algorithm has the property that the input point P_1 can have the same Z-coordinate as $P_1 + P_2$ at the end of the addition. The gain of this formula is less than what is gained with the simplified formula on $\mathbb{F}_p$. However, it can be efficiently used with our proposed scalar multiplication algorithm (see Section 4).

We present in the next Section scalar multiplication algorithms resistant against side-channel attacks on both $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$.

3 Classical side-channel resistant scalar multiplication algorithms

Scalar multiplication, or point multiplication, is an operation that computes $[k]P$ where k is an integer and P is an elliptic curve point. The square-and-multiply is a well-known method for exponentiation. The additive version of this algorithm, called double-and-add (Algorithm 1), can be used as a basic scalar multiplication technique.

Algorithm 1: Left-to-right double-and-add

input : $P \in E$ and $k = (k_{n-1} \dots k_1 k_0)_2$
output: $[k]P \in E$

```
1 Q ← P
2 for i ← n − 2 to 0 do
3 |   Q ← [2]P
4 |   if k_i = 1 then
5 |   |   Q ← Q + P
6 return Q
```

With standard addition and doubling formulas, an attacker can detect bit information on the scalar k by SPA [Cor99]. The power consumption traces of an addition and a doubling are different enough to be distinguished. Coron proposed in 1999 a dummy addition method [Cor99], also known as double-and-always-add, which represents the simplest algorithm of this type (Algorithm 2).

Yoon et al. [YJL03] propose a sort of double-and-always-add algorithm for elliptic curves over binary fields in affine coordinates. In order to avoid the performance drawback of the affine coordinates they design an inverter architecture

One can also apply different methods to convert any scalar multiplication algorithm, as the basic double-and-add, into a SPA-resistant version. Chevallier-Mames et al. [CMCJ04]

Algorithm 2: Double-and-always-add

input : $P \in E$ and $k = (k_{n-1} \dots k_1 k_0)_2$
output: $[k]P \in E$

1 $Q_0 \leftarrow P$
2 **for** $i \leftarrow n-2$ **to** 0 **do**
3 $\quad Q_0 \leftarrow [2]Q_0$
4 $\quad Q_1 \leftarrow Q_0 + P$
5 $\quad Q_0 \leftarrow Q_{k_i}$ `/* ` Q_{k_i} `equals either` Q_0 `or` Q_1 `*/`
6 **return** Q_0

proposed the idea of side-channel atomicity. Each elliptic curve operation is implemented as the repetition of blocks of instructions that look alike in the power trace. The code of the scalar multiplication algorithm is then unrolled such that it appears as a repetition of the same atomic block. The sequence of blocks does not depend on the scalar used and their algorithm is then secure against SPA.

Another approach to SPA resistance is using indistinguishable addition and doubling algorithms in the scalar multiplication [CJ01, BDJ04]. Jacobi form, Hesse form or Edwards form elliptic curves allow the same algorithm for both additions and doublings. However, we only consider in this paper standardized curves recommended by specifications [X9.98, NIS00, SEC00]. Brier et al. [BDJ04] proposed a unified addition and doubling formula that costs $16M + 3S$ in $\mathbb{F}_p$ and $20M + 3S$ on $\mathbb{F}_{2^m}$ for projective representation. These two techniques can either be applied on elliptic curve over $\mathbb{F}_p$ or $\mathbb{F}_{2^m}$. However these two approaches only offer SPA resistance, FA attacks would still be a threat.

Algorithm 3: Montgomery ladder

input : $P \in E$ and $k = (k_{n-1} \dots k_1 k_0)_2$
output: $[k]P \in E$

1 $P_0 \leftarrow P$
2 $P_1 \leftarrow [2]P$
3 **for** $i \leftarrow n-2$ **to** 0 **do**
$\quad$ `//` $k_i =$ `either 0 or 1 and` $\bar{k_i} = 1 - k_i$
4 $\quad P_{\bar{k_i}} \leftarrow P_0 + P_1$
5 $\quad P_{k_i} \leftarrow [2]P_{k_i}$
6 **return** P_0

Finally, we consider the Montgomery ladder algorithm (Algorithm 3) which was originally proposed in [Mon87] only for Montgomery-type elliptic curves. Montgomery's original idea was based on the fact that the sum of two points whose difference is a known point can be computed without the y-coordinate of the two points. In [BJ02], Brier and Joye generalized the algorithm to any elliptic curves over $\mathbb{F}_p$. Their adaptation requires $9M + 2S$ for an addition and $6M + 3S$ for a doubling. The complexity of this general algorithm is then $n(15M + 5S) + 3M + S + I$ for a n-bit scalar, where I is a modular

inversion in the field $\mathbb{F}_p$ and $3M + S + I$ is the cost to recover the Y-coordinate at the end. We can also note Izu and Takagi work [IT02] that, at the same moment as Brier and Joye, also generalized Montgomery's ladder. They obtained slightly better results with a complexity of $n(13M + 4S) + 11M + 2S$ for a n-bit scalar.

In the $\mathbb{F}_{2^m}$ case, López et al. [LD99] generalized Montgomery's idea with an algorithm that only requires $n(6M + 5S) + 1I + 10M + 1S$ for a n-bit scalar.

Since the Montgomery ladder is, by construction, an interesting algorithm for side-channel resistance (see Section 5) we use it as a basis for our multiplication methods. However, we can't use classical doublings with Meloni's addition formula in a point scalar multiplication algorithm as, for each bit, we would need to compute $[2]P_{k_i}$ (Algorithm 3, Line 5) so that it has the same Z-coordinate as $P_{\bar{k}_i} = P_0 + P_1$ (Algorithm 3, Line 4). We would lose the benefit of the simplified addition. Meloni proposed a Fibonacci-and-add algorithm [Mel07] that performed scalar multiplication only using his addition formula. The gain of the addition is counteracted by a representation of the scalar k that is much larger than its binary representation. By modifying the Montgomery ladder structure, we are able to only use Meloni's additions while using the binary representation of k.

4 Our side-channel resistant algorithms alternatives

Let R, a n-bit integer, be the order of the elliptic curve point P, and let $k < R - 1$ an integer.

4.1 Modified Montgomery ladder algorithm

In order to use efficiently the simplified addition we modify the Montgomery ladder structure (Algorithm 4). The algorithm now only uses point additions. However, simplified additions cannot be used yet as the two input points need the same Z-coordinate.

Algorithm 4: Montgomery ladder with additions

input : $P \in E$ and $k = (k_{n-1} \dots k_1 k_0)_2$
output: $[k]P \in E$

1 $P_1 \leftarrow P$;
2 $P_2 \leftarrow [2]P$;
3 **for** $i \leftarrow n - 2$ **to** 0 **do**
4 $\quad P_1 \leftarrow P_1 + P_2$;
5 $\quad P_2 \leftarrow P_1 + (-1)^{\bar{k}_i} P$;
6 **return** P_1

4.2 Tweaking simplified addition algorithms

In order to use simplified additions, we must have $Z_{P_2} = Z_{P_1}$ at the end of each round in order to add them in the next one. Fortunately, this is a property of the simplified addition. However, we also need that the point $\pm P$ has the same Z-coordinate as P_1 before computing $P_2 \leftarrow P_1 + (-1)^{\bar{k}_i} P$ (Algorithm 4, Line 5).

	$\mathbb{F}_p$	$\mathbb{F}_{2^m}$
`SimpleAdd`	$5M + 2S$	$7M + 2S$
`SimpleAddSub`	$6M + 3S$	$11M + 2S$

Tab. 1: Complexity in field operations of the different simplified addition algorithms.

We propose to recompute the point P at each round within a modified simplified addition algorithm that computes both addition and subtraction so that the points have the same Z-coordinate at the end of the function. The algorithm is called `SimpleAddSub`,

$$\texttt{SimpleAddSub} \rightarrow (\tilde{P}_1, P_1 + P_2, P_1 - P_2) \text{ with } Z_{\tilde{P}_1} = Z_{P_1+P_2} = Z_{P_1-P_2}.$$

Let $P_1 = (X_1, Y_1, Z)$, $P_2 = (X_2, Y_2, Z)$ both unequal to ∞ and $P_2 \neq \pm P_1$. Let $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$, $P_4 = P1 - P2 = (X_4, Y_4, Z_4)$ and $\tilde{P}_1 = (\tilde{X}_1, \tilde{Y}_1, \tilde{Z}_1)$. The algorithm `SimpleAddSub` on $\mathbb{F}_p$ is computed as:

$$\begin{aligned} A &= (X_2 - X_1)^2, & B &= X_1 A, & C &= X_2 A, \\ D &= (Y_2 - Y_1)^2, & E &= (-Y1 - Y2)^2, & F &= Y_1(C - B). \end{aligned}$$

$$\begin{aligned} \tilde{X}_1 &= B, & \tilde{Y}_1 &= F, & \tilde{Z}_1 &= ZA, \\ X_3 &= D - B - C, & Y_3 &= (Y_2 - Y_1)(B - X_3) - F, & Z_3 &= ZA, \\ X_4 &= E - B - C, & Y_4 &= (B - X_4)(-Y_1 - Y_2) - F, & Z_4 &= ZA. \end{aligned}$$

The algorithm `SimpleAddSub` on $\mathbb{F}_{2^m}$ is computed as:

$$\begin{aligned} A &= X_1 + X_2, & B &= A^2, & C &= AB, & D &= Y_1 + Y_2, \\ D' &= ZX_1 + D, & E &= CY_2, & F &= BX_2, & G &= D + Z_3, \\ G' &= D' + Z_3, & H &= FD, & H' &= FD', & I &= aZ_3^2. \end{aligned}$$

$$\begin{aligned} \tilde{X}_1 &= F, & \tilde{Y}_1 &= E, & \tilde{Z}_1 &= ZA, \\ X_3 &= I + DG + C, & Y_3 &= X_3 G + E + H, & Z_3 &= ZA, \\ X_4 &= I + D'G' + C, & Y_4 &= X_4 G' + E + H', & Z_4 &= ZA. \end{aligned}$$

We summarize the complexities of the simplified addition algorithms in Table 1.

4.3 Combining Montgomery ladder-like multiplication with `SimpleAddSub`

We have to differentiate the $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$ cases. If $P = (X, Y, Z) \in E(\mathbb{F}_{2^m})$ is in Jacobian projective coordinates then $-P = (X, ZX + Y, Z)$. Whereas, if $P = (X, Y, Z) \in E(\mathbb{F}_p)$ is also in Jacobian projective coordinates then $-P = (X, -Y, Z)$. Because of the use of the Z-coordinate in the $\mathbb{F}_{2^m}$ case for this operation, we are not able to fully optimize the scalar multiplication algorithm.

We first introduce a basic version of our proposed multiplication algorithm, called `BasicScalarMult` (Algorithm 5), that combines a Montgomery ladder-like structure and

the `SimpleAddSub`. We note $Q[0]$, $Q[1]$ and $Q[2]$ respectively the outputs of `SimpleAddSub` $\tilde{P_1}$, $P_1 + P_2$ and $P_1 - P_2$ (Algorithm 6 lines 4 and 7). At each round, line 6, the algorithm will get an updated point P with the correct Z-coordinate thanks to the added subtraction in `SimpleAddSub`. Also, after the second `SimpleAddSub`, we always have, if $P_1 = [r]P$, then $P_2 = [r-1]P$. Hence, in the next round, line 6, we again get an updated $P = P_1 - P_2$.

Algorithm 5: `BasicScalarMult`

input : $P \in E$ and $k = (k_{n-1} \dots k_1 k_0)_2$
output: $[k]P \in E$

1 $P_1 \leftarrow [2]P$
2 $P_2 \leftarrow P$
// We assume $Z_{P_1} = Z_{P_2}$
3 **for** $i \leftarrow n-2$ **to** 0 **do**
4 $\quad Q \leftarrow$ `SimpleAddSub`(P_1, P_2)
5 $\quad P_1 \leftarrow Q[1]$ /* $P_1 \leftarrow (P_1 + P_2)$ */
6 $\quad P_2 \leftarrow Q[2]$ /* $P_2 \leftarrow (P_1 - P_2) = P$ */
7 $\quad Q \leftarrow$ `SimpleAddSub`(P_1, P_2)
8 $\quad P_1 \leftarrow Q[k_i]$ /* $P_1 \leftarrow \tilde{P_1}$ or $P_1 \leftarrow P_1 + P_2$ */
9 $\quad P_2 \leftarrow Q[2\bar{k}_i]$ /* $P_2 \leftarrow \tilde{P_1}$ or $P_2 \leftarrow P_1 - P_2$ */
10 **return** P_2

The `BasicScalarMult` scalar multiplication only uses the `SimpleAddSub` algorithm. Depending on the type of finite field used, the complexities are

- on $\mathbb{F}_{2^m}$: $n(22M + 4S)$,
- on $\mathbb{F}_p$: $n(12M + 6S)$,

where n is the size in bits of the scalar.

In $\mathbb{F}_p$, we can further improve the performance of our algorithm if we note that within the loop of the scalar multiplication, the Z-coordinate of the points is not used. We simplify the `SimpleAddSub` algorithm on $\mathbb{F}_p$ into a `SimpleAddSubWoZ` version without computing the Z-coordinate. Its complexity is then $5M + 3S$.

We propose a second version of our scalar multiplication algorithm called, `OptScalarMult` (Algorithm 6), using `SimpleAddSubWoZ`. Because the Z-coordinate is not computed inside the main loop, the final Z is retrieved in the last round for minimal computational costs. This optimized version has now a complexity of $n(10M + 6S)$. More details on this $\mathbb{F}_p$ improvement can be found in [VD10].

4.4 Efficiency evaluation

Our goal in this article is to propose scalar multiplication algorithms resistant against SPA and FA attacks, patent-free, but also efficient enough. The Montgomery ladder structure is one of the best regarding side-channel resistance, notably against FA. Unfortunately, its well-known variant where one computes the scalar multiplication without using the y-coordinate is patented [VMAG99] (US Patent number: 6782100). An alternative is to

Algorithm 6: `OptScalarMult`

input : $P \in E(\mathbb{F}_p)$ and $k = (k_{n-1} \dots k_1 k_0)_2$
output: $[k]P \in E(\mathbb{F}_p)$

1 $P_1 \leftarrow [2]P$
2 $P_2 \leftarrow P$
// We assume $Z_{P_1} = Z_{P_2}$
3 $P_{save} \leftarrow P$
4 **for** $i \leftarrow n-2$ **to** 1 **do**
5 $\quad Q \leftarrow$ `SimpleAddSubWoZ`(P_1, P_2)
6 $\quad P_1 \leftarrow Q[1]$ /* $P_1 \leftarrow (P_1 + P_2)$ */
7 $\quad P_2 \leftarrow Q[2]$ /* $P_2 \leftarrow (P_1 - P_2) = P$ */
8 $\quad Q \leftarrow$ `SimpleAddSubWoZ`(P_1, P_2)
9 $\quad P_1 \leftarrow Q[k_i]$ /* $P_1 \leftarrow \tilde{P_1}$ or $P_1 \leftarrow P_1 + P_2$ */
10 $\quad P_2 \leftarrow Q[2\bar{k}_i]$ /* $P_2 \leftarrow \tilde{P_1}$ or $P_2 \leftarrow P_1 - P_2$ */
// Last round
11 $Q \leftarrow$ `SimpleAddSubWoZ`(P_1, P_2)
12 $P_1 \leftarrow Q[1]$ /* $P_1 \leftarrow (P_1 + P_2)$ */
13 $P_2 \leftarrow Q[2]$ /* $P_2 \leftarrow (P_1 - P_2) = P$ */
// Compute Z_P
14 $Z_{final} \leftarrow X_{P_2} \cdot Y_{P_{save}}$
15 $Z_{final} \leftarrow (Z_{final})^{-1}$
16 $Z_{final} \leftarrow Z_{final} \cdot Y_{P_2}$
17 $Z_{final} \leftarrow Z_{final} \cdot X_{P_{save}}$
18 $Z_{final} \leftarrow Z_{final} \cdot Z_{P_{save}}$
19 $Z_{final} \leftarrow (Z_{final} \cdot (X_{P_2} - X_{P_1}))$
20 $Q \leftarrow$ `SimpleAddSubWoZ`(P_1, P_2)
21 $P_1 \leftarrow Q[k_i]$ /* $P_1 \leftarrow \tilde{P_1}$ or $P_1 \leftarrow P_1 + P_2$ */
22 $P_2 \leftarrow Q[2\bar{k}_i]$ /* $P_2 \leftarrow \tilde{P_1}$ or $P_2 \leftarrow P_1 - P_2$ */
23 $P_2 \leftarrow [X_{P_2}, Y_{P_2}, Z_{final}]$
24 **return** P_2

$\mathbb{F}_p$	Complexity (per bit of scalar)
GML	$12M + 13S \approx 25M$
Brier et al.[BJ02]	$15M + 5S \approx 20M$
Izu et al. [IT02]	$13M + 4S \approx 17M$
`BasicScalarMult`	$12M + 6S \approx 18M$
`OptScalarMult`	$10M + 6S \approx 16M$
$\mathbb{F}_{2^m}$	Complexity (per bit of scalar)
GML	$18M + 10S \approx 28M$
López et al.[LD99]	$6M + 5S \approx 11M$
`BasicScalarMult`	$22M + 4S \approx 26M$

Tab. 2: Summary of side-channel resistant scalar multiplication algorithms.

implement the Montgomery ladder structure using basic point addition and point doubling formulas instead of the y-free ones. The loss in efficiency is important however the patent issues are avoided. We call this method Generic Montgomery Ladder (GML). On $\mathbb{F}_p$, our propositions give better performances than Brier et al. [BJ02] and Izu et al. [IT02] ones where the y-coordinate is not computed. It also obviously outperforms a GML implementation. On $\mathbb{F}_{2^m}$, even if our `BasicScalarMult` algorithm is less efficient than López et al. [LD99] y-free Montgomery ladder, it offers an alternative to this patented method and an improvement compared to a generic implementation. Table 4.4 gives a comparison of the different scalar multiplication algorithms.

5 Resistance against side-channel attacks

As previously stated, our main goal is to use a scalar multiplication algorithm that is efficient and secure against classical Side-Channel Attacks (SCA), particularly SCA using power consumption traces. We can differentiate three main categories of SCA: Simple Power Analysis (SPA), Differential Power Analysis (DPA) and Fault Analysis (FA). They are a real threat on embedded devices and have to be taken into account when one chooses an algorithm to implement. We briefly review the SCA resistance of scalar multiplication algorithms and the available countermeasures.

Simple Power Analysis. Standard double-and-add algorithms, like Algorithm 1, contain conditional branching where different instructions are executed depending on the bit values of the scalar. The two branches behave differently and this translates to a change of side-channel information being leaked by the device. With SPA-like attacks, an attacker can easily distinguish bit values. Therefore, algorithms with dummy operations, like double-and-always-add (Algorithm 2), were proposed. The conditional branching now contains the same operations by adding dummy operations to equalize the side-channel leakage. The Montgomery ladder is highly regular as it computes, for each bit regardless of its value, a doubling and an addition. Our multiplication algorithms are based on a modified Montgomery ladder. Each of our algorithms computes the same sequence of instructions regardless of the value the bit of the scalar takes. The computations are a fixed pattern unrelated to the bit information of the scalar. The side-channel information also becomes a fixed pattern. Thus, SPA-like attacks are defeated.

Differential Power Analysis. Differential side-channel analysis estimates the value of an intermediate result of the algorithm using statistical tools. DPA-like attacks need a so-called leakage function that computes for each input message the hypothetical power consumption of a targeted intermediate value that also depends on the value of the secret. The guessed consumption is then compared to the actual power consumption trace of the device in order to find a statistical relation. SPA-resistance does not imply DPA-resistance of an algorithm. However, our proposed SPA-resistant algorithms are easy to enhance. Countermeasures against DPA aim to make impossible the guessing of the leakage function output by using random numbers. A lot of randomization methods have been proposed for elliptic curve cryptosystems. Coron in [Cor99] proposed representing elliptic curve points using randomized projective coordinates. Let $P = (x, y, z)$ be a point in Jacobian projective coordinates. Then for all non-zero integers r, (r^2x, r^3y, rz) represents the same point. Only knowing the point P, the bit sequence of the randomized point is so different to P that statistical tools of DPA can't find relationships. The additional computational cost is $4M + 1S$ at the beginning of the scalar multiplication. Joye and Timen [JT01] proposed the use of randomized isomorphisms between elliptic curves. A point $P = (x, y)$ is randomized into $(r^{-2}x, r^{-3}y, 1)$ in Jacobian coordinates for an non-zero integer r, with elliptic curve parameters $a' = r^{-4}a$ and $b' = r^{-6}b$. The advantage of this method is that the Z-coordinate of the randomized point is 1. Hence, optimizations in the elliptic curve algorithms can be applied. However, Joye-Tymen randomization requires more additional storage than Coron's. The initial transformation of the point requires $4M + 2S$ plus the storage of two field elements. We can also briefly mention other randomization techniques against DPA. Coron [Cor99] introduced the randomized exponent method, as well as the randomized base point. Clavier and Joye [CJ01] proposed splitting the scalar k into r and $k - r$, with r a random integer. One then computes $[k]P$ as $[k - r]P + [r]P$.

Fault Analysis. Fault attacks are based on the fact that a fault during a cryptographic computation leads to a faulty result. If the device does not detect the fault and does not prevent the output, an attacker can exploit the results. Using knowledge of faulty results, correct ones and the precise place of induced faults, an attacker can recover bits of a secret. Numerous mechanisms for fault injection have been discovered and researched [HCN+04]. Double-and-always-add algorithms are obviously susceptible to fault attacks. As previously seen, the algorithm runs in constant time, the same operations are computed regardless of bit values. Hence, an attacker can easily detect the operations in Algorithm 2, lines 3 and 4. If, for example, k_i equals 0, and the adversary injects a fault in the computation of Q_1, this intermediate result is a dummy operation and the final result of the multiplication has not changed. Therefore, the attacker knows that $k_i = 0$ because his fault had no effect on the final result. By repeating this technique, he can recover the secret scalar. This type of fault injection is also called computational safe-error attack. However, for the Montgomery ladder, the situation is different as every intermediate result is used to compute the final result. Hence, if the attacker induces a fault the final result will inevitably be corrupted [JY02]. This type of fault attack is called computational safe-error attack or C safe-error attack [YKLM02]. Recently, Fouque et al. [FLRV08] presented the twist curve attacks: a powerful fault attack against a Montgomery ladder implementation using no y-coordinate. However, for our case, the y-coordinate is used in all our propositions.

Using the Montgomery ladder structure, our proposed scalar multiplication algorithms are SPA and FA resistant. In order to thwart DPA attacks, the countermeasure, proposed by Clavier et al. [CJ01], that consists in the random splitting of the scalar can be applied to our algorithm. Finally, we can add point verification [BMM00], that checks if a point lies on a curve or not, once the scalar multiplication is completed. With this set of countermeasures, we can be relatively confident about the SCA resistance of our algorithms against known attacks.

6 Conclusion

We present in this article a new scalar multiplication method using Meloni's technique on both $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$. We obtain alternatives to the patented Montgomery ladder that are as resistant against SCA and that are efficiently competitive. The SCA resistance against SPA, and more importantly FA, is derived from the Montgomery ladder structure. The DPA resistance can be easily added using one of the well-known countermeasures proposed for elliptic curve cryptosystems. Furthermore, we present algorithms that are computationally efficient. Even if on $\mathbb{F}_{2^m}$, López et al. [LD99] proposition is more efficient, our alternative is better than a generic Montgomery ladder implementation while avoiding patent issues related to y-free implementations. On $\mathbb{F}_p$, our method is even faster than previous Montgomery ladder propositions that don't use the y-coordinate, thus providing a fast SPA and FA resistant scalar multiplication algorithm.

References

[BDJ04] E. Brier, I. Déchène, and M. Joye. Unified point addition formulæ for elliptic curve cryptosystems. *Embedded Cryptographic Hardware: Methodologies and Architectures. Nova Science Publishers*, pages 247–256, 2004.

[BJ02] E. Brier and M. Joye. Weierstraß elliptic curves and side-channel attacks. In *PKC 2002, LNCS*, pages 335–345, 2002.

[BL07] J. D. Bernstein and T. Lange. Explicit-formulas database, 2007. http://www.hyperelliptic.org/EFD.

[BMM00] I Biehl, B. Meyer, and V. Müller. Differential fault attacks on elliptic curve cryptosystems. *CRYPTO 2000, LNCS*, 1880:131–146, 2000.

[CJ01] C. Clavier and M. Joye. Universal exponentiation algorithm a first step towards provable spa-resistance. *CHES 2001, LNCS*, 2162:300–308, 2001.

[CJ05] M. Ciet and M. Joye. Elliptic curve cryptosystems in the presence of permanent and transient faults. *Designs, Codes and Cryptography*, 36:33–43, 2005.

[CMCJ04] B. Chevallier-Mames, M. Ciet, and M. Joye. Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. *IEEE Transactions on Computers*, 53:760–768, 2004.

[Cor99] J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. *CHES 1999, LNCS*, 1717:292–302, 1999.

[FLRV08] P-A Fouque, R Lercier, D Réal, and F Valette. Fault attack on elliptic curve montgomery ladder implementation. In *Proceedings of FDTC 2008*, pages 92–98, 2008.

[HCN+04] H. B.-E. Hamid, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The sorcerer's apprentice guide to fault attacks. Cryptology ePrint Archive, Report 2004/100, 2004. http://eprint.iacr.org/2004/100.

[IT02] T. Izu and T. Takagi. A fast parallel elliptic curve multiplication resistant against side channel attacks. *PKC 2002, LNCS*, 2274:371–374, 2002.

[JT01] M. Joye and C. Tymen. Protections against differential analysis for elliptic curve cryptography. *CHES 2001, LNCS*, 2162:377–390, 2001.

[JY02] M. Joye and S.M. Yen. The montgomery powering ladder. *CHES 2002, LNCS*, 2523:1–11, 2002.

[LD99] J. López and R. Dahab. Fast multiplication on elliptic curves over GF(2^m) without precomputation. *CHES 1999, LNCS*, 1717:316–328, 1999.

[Mel07] N. Meloni. New point addition formulae for ecc applications. *Arithmetic of Finite Fields, LNCS*, 4547:189–201, 2007.

[Mon87] P.L. Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48:243–264, 1987.

[NIS00] NIST. Recommended elliptic curves for federal government use, appendix to FIPS 186-2, 2000.

[SEC00] SEC2. Standards for Efficient Cryptography Group/Certicom Research. Recommanded Elliptic Curve Cryptography Domain Parameters, 2000.

[VD10] A. Venelli and F. Dassance. Faster side-channel resistant elliptic curve scalar multiplication. In *Conference on Arithmetic, Geometry, Cryptography and Coding Theory, AMS-CONM*, 2010.

[VMAG99] S. Vanstone, R. Mullin, A. Antipa, and R. Gallant. Accelerated finite field operations on an elliptic curve. Patent, 1999. International Patent Publication, WO/1999/049386.

[X9.98] ANSI X9.62. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). Cornell University, Research Report, 1998.

[YJL03] J.C. Yoon, S.W. Jung, and S. Lee. Architecture for an Elliptic Curve Scalar Multiplication Resistant to Some Side-Channel Attacks. *Information Security and Cryptology-ICISC 2003, LNCS*, 2971:139–151, 2003.

[YKLM02] S.M. Yen, S. Kim, S. Lim, and S. Moon. A countermeasure against one physical cryptanalysis may benefit another attack. *Information Security and Cryptology - ICISC 2001, LNCS*, 2288:414–427, 2002.