Faster Side-Channel Resistant Elliptic Curve Scalar Multiplication

Alexandre VENELLI and François DASSANCE

Abstract. We present a new point scalar multiplication algorithm on classical Weierstrass elliptic curves over fields of characteristic greater than 3. Using Meloni's formula that efficiently adds two points with the same Z-coordinates, we develop an algorithm computing $[k]P$ only with these point additions. We combine Meloni's addition with a modified version of a Montgomery ladder, a well-established side-channel resistant method for scalar multiplication. Our aim is to construct an algorithm that is resistant, by construction, against Simple Power Analysis (SPA) and Fault Analysis (FA) while still being efficient. We present four versions of our algorithm with various speed-ups depending on the available memory of the device. Finally, we compare our method with state-of-the-art algorithms at the same level of side-channel resistance.

1. Introduction

Smart cards and more generally low powered computational devices, need efficient algorithms which must be resistant to side-channel analysis. Side-channel attacks use information observed during the execution of the algorithm to determine the secret key. The two main classes of side-channel attacks are: simple side-channel attacks, like Simple Power Analysis (SPA), which analyze the trace of a single execution of the algorithm, and differential side-channel attacks, like Differential Power Analysis (DPA), which compare the traces of multiple executions. Another kind of implementation attacks are Fault Attacks (FA). Initially reported on RSA, they were quite naturally extended to other group based crytosystems. Biel, Meyer and Mller [**BMM00**] showed how to exploit errors in elliptic curve scalar multiplications. Their results were extended by Ciet and Joye [**CJ05**].

Elliptic curve (EC) cryptosystems are of great interest because they require less memory and hardware ressources than other cryptographic standards like RSA for a given security level. They are considered particularly suitable for implementation on smart cards and mobile devices. Because of the physical characteristics of these devices and their use in potentially hostile environments, they are particularly sensitive to side-channel attacks. The most important operation in EC cryptosystems is the point scalar multiplication $[k]P$. Its computational cost is decisive in the

2010 *Mathematics Subject Classification.* 14H52, 65Y10.

overall efficiency of the EC algorithms but securing it can be very time consuming. Numerous articles in the literature deal with securing the scalar multiplication against different side-channel attacks.

We propose a new scalar multiplication algorithm that overcomes both the efficiency and the side-channel resistance problems. We use Meloni's addition formula that is very efficient but requires the two input points to have the same Z-coordinate. Modifying the Montgomery ladder algorithm, we obtain an algorithm that uses only Meloni's addition and that is resistant against SPA and FA like Montgomery's algorithm.

This paper 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 faster multiplication algorithms. Finally, Section 5 analyzes the security against side-channel attacks of our algorithm and compares its efficiency with other methods at the same level of side-channel resistance.

2. Elliptic curve arithmetic

We consider elliptic curves defined over $K = \mathbb{F}_p$, with $p > 3$, a finite field of p elements. An elliptic curve E over a field K is defined by an equation of the form:

$$E/K : y^2 = x^3 + ax + b$$

where $a, b \in K$ satisfy $\Delta = 4a^3+27b^2 \neq 0 \bmod p$. The set of all the points on E with the point at infinity, denoted ∞, is equipped with an additive group structure. The coordinate system chosen for a point addition or doubling is very important in terms of efficiency. One can look at [**BL07**] for a summary of addition and doubling's complexity in different coordinate systems. In practice, the Jacobian coordinates are often used because they offer a great compromise between computational costs and memory usage. A point P in Jacobian coordinates is noted $P = (X, Y, Z)$ and represents the affine point $(\frac{X}{Z^2}, \frac{Y}{Z^3})$. Classical addition and doubling formulas [**BL07**] are as follows:

Point doubling. Let $P = (X, Y, Z)$, $P_3 = [2]P = (X_3, Y_3, Z_3)$ and suppose $P \neq -P$.

$$A = X^2, \quad B = Y^2, \quad C = B^2, \quad D = Z^2, \quad E = 2((X + B)^2 - A - C),$$

$$F = 3A + aD^2, \quad G = F^2 - 2E$$

$$\begin{cases} X_3 &= G, \\ Y_3 &= F(E - G) - 8C, \\ Z_3 &= (Y + Z)^2 - B - D. \end{cases}$$

A point doubling can be done with 1 multiplications and 8 squarings in the field K, noted $1M + 8S$.

Point addition. Let $P_1 = (X_1, Y_1, Z_1)$, $P_2 = (X_2, Y_2, Z_2)$ both unequal to ∞ and $P_2 \neq \pm P_1$. Let $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$.

$$A = Z_1^2, \quad B = Z_2^2, \quad C = X_1B, \quad D = X_2A, \quad E = Y_1Z_2B, \quad F = Y_2Z_1A,$$

$$G = D - C, \quad H = (2G)^2, \quad I = GH, \quad J = 2(F - E), \quad K = CH$$

$$\begin{cases} X_3 &= J^2 - I - 2K, \\ Y_3 &= J(K - X_3) - 2EI, \\ Z_3 &= ((Z_1 + Z_2)^2 - A - B)G. \end{cases}$$

A general point addition costs $11M + 5S$.

We use in our point scalar multiplication algorithm the simplified addition formula found by Meloni [**Mel07**]. If $P_1 = (X_1, Y_1, Z)$ and $P_2 = (X_2, Y_2, Z)$ are two points in Jacobian coordinates with the same Z-coordinate, the following formula can be applied:

Simplified point addition. 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 $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$.

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

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

The point addition in this special case only costs $5M + 2S$. It is even faster than the general point doubling in Jacobian coordinates. In this state, the algorithm is not very useful because it is unlikely for both P_1 and P_2 to have the same Z-coordinate. 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. He calls this algorithm

$$\texttt{NewAdd}(P_1, P_2) \to (\tilde{P_1}, P_1 + P_2).$$

NewAdd. 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 $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$.

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

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

and

$$\begin{cases} X_1 &= B, \\ Y_1 &= E, \\ Z &= Z_3. \end{cases}$$

Meloni also shows that the classical doubling can be modified so that it returns $\tilde{P}$ and $[2]P$ with same Z-coordinate without adding computational cost.

3. Classical side-channel resistant scalar multiplication algorithms

A standard method for performing the scalar multiplication $[k]P$ is the left-to-right double-and-add algorithm (Algorithm 1). It is the elliptic curve equivalent of the square-and-multiply for exponentiation in finite fields. Let k be a positive integer and P a point on an elliptic curve. Let

$$k = k_{n-1}2^{n-1} + \cdots + k_1 2^1 + k_0 2^0$$

be the binary representation of k where $k_{n-1} = 1$. We can compute $[k]P$ as follows with the left-to-right double-and-add algorithm.

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 $Q \leftarrow P$;
2 for $i \leftarrow n-2$ to $0$ do
3     $Q \leftarrow [2]P$;
4     if $k_i = 1$ then
5         $Q \leftarrow Q + P$;
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).

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     $Q_0 \leftarrow [2]Q_0$;
4     $Q_1 \leftarrow Q_0 + P$;
5     $Q_0 \leftarrow Q_{k_i}$            /* $Q_{k_i}$ equals either $Q_0$ or $Q_1$ */;
6 return $Q_0$
```

Chevallier-Mames et al. [**CMCJ04**] 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. A doubling in Jacobian coordinates is computed using 10 atomic blocks and 16 blocks for an addition, each atomic block costing $1M$. However their construction uses dummy operations and can then be sensitive to fault attacks.

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 recommanded by specifications [**X9.98, NIS00, SEC00**]. Brier et al. [**BDJ04**] proposed a unified addition and doubling formula for generic Weierstraß curves that cost $16M + 3S$ for Jacobian coordinates. One of the benefits of this type of countermeasure is that there is no use of dummy operations, hence fault analysis techniques cannot be used.

We can also mention the NAF-based multiplication algorithms [**JY00, OT04**]. The non-adjacent (NAF) form is a unique signed digit representation of an integer using the digits $\{-1, 0, 1\}$, such that no two adjacent digits are both non-zeros. NAF algorithms take advantage of the fact that negating a point on an elliptic curve simply requires a change in the sign of the Y-coordinate, substractions are cheap operations. However classical NAF multiplications can be sensitive to sign change fault attacks [**BOS06**]. Recently, the authors of [**GLS09**] and [**LG09**] pointed out the use of Meloni's formulas for the purpose of precomputations in NAF-based multiplication algorithms.

Finally, we consider the Montgomery ladder algorithm (Algorithm 3) which was originally proposed in [**Mon87**] only for Montgomery-type elliptic curves. In [**BJ02**], Brier and Joye generalized the algorithm to any elliptic curves in short Weierstraß equations. 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. His algorithm is very efficient on a certain family of elliptic curves, called Montgomery's curves. In this case, the differential addition costs $4M + 2S$ and the doubling $2M + 2S + 1D$ where $1D$ is a multiplication by a constant. Brier and Joye's 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.

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
4      $P_{\bar{k_i}} \leftarrow P_0 + P_1$;
5      $P_{k_i} \leftarrow [2]P_{k_i}$;
6  return $P_0$
```

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. 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 multiplication

Let R, a n-bit integer, be the order of the elliptic curve point P, and let $k < R-1$ an integer. We use in our approach a modified version of the Montgomery ladder (Algorithm 4) with Meloni's addition to construct a multiplication algorithm resistant to both SPA and FA (see Section 5). However, as previously stated, Meloni's formula needs as input two points with the same Z-coordinate. We both describe a naive method and our proposed solution to deal with this issue.

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     $P_1 \leftarrow P_1 + P_2$;
5     $P_2 \leftarrow P_1 + (-1)^{\bar{k}_i} P$;
6 return $P_2$
```

4.1. A naive approach to the Z-coordinate problem. 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. Fortunately, this is a property of the `NewAdd` algorithm. Also, the point $\pm P$ must have the same Z-coordinate as P_1 before computing $P_2 \leftarrow P_1 + (-1)^{\bar{k}_i} P$ (Algorithm 4, Line 5). We could recalculate an updated P at each round with $Z_P = Z_{P_1}$ but we would need to:

(1) Store the point $P = (X, Y, Z)$ during the whole scalar multiplication.
(2) Compute and store the modular inversion Z^{-1} at the beginning of the algorithm.
(3) Compute, at each round, if $P = (X, Y, Z)$ and $P_1 = (X_1, Y_1, Z_1)$, the integer $\lambda = Z_1 Z^{-1}$. Finally, we would have $P' = \pm(\lambda X, \lambda Y, \lambda Z)$ for a total of $4M$.

For a n-bit scalar k, the cost of a multiplication $[k]P$ will be $n(2(5M + 2S) + 4M + S) + I = n(14M + 5S) + I$ where I is the cost of an inversion in $\mathbb{F}_p$.

4.2. Updating P's coordinates more efficiently. We propose to recompute the point P at each round within a modified addition algorithm (Algorithm 5), with an appropriate Z-coordinate. We call

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

In `NewAddSub` we take the simplified addition and we add the subtraction for additional cost $1M + 1S$ in time. Finally our `NewAddSub` costs $6M + 3S$ where `NewAdd` costs $5M + 2S$.

We can now write a point scalar multiplication algorithm called `FullMult` (Algorithm 6). We note $Q[0]$, $Q[1]$ and $Q[2]$ respectively the outputs of `NewAddSub` $\tilde{P_1}$, $P_1 + P_2$ 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 substraction in `NewAddSub`. Also, after the second `NewAddSub`, we always

Algorithm 5: `NewAddSub`

```
input  : $P_1 = (X_1, Y_1, Z)$ and $P_2 = (X_2, Y_2, Z)$
output: $(\tilde{P}_1, P_1 + P_2, P_1 - P_2)$
1  $R_1 \leftarrow X_2 - X_1$;
2  $Z \leftarrow Z \cdot R_1$                    /* Final $Z$ */;
3  $R_1 \leftarrow R_1^2$;
4  $X_1 \leftarrow X_1 \cdot R_1$                /* $X_{\tilde{P}_1}$ */;
5  $X_2 \leftarrow X_2 \cdot R_1$;
6  $R_1 \leftarrow Y_2 - Y_1$;
7  $R_2 \leftarrow R_1^2$;
8  $R_2 \leftarrow R_2 - X_1 - X_2$             /* $X_{P_1+P_2}$ */;
9  $R_3 \leftarrow X_1 - R_2$;
10 $R_3 \leftarrow R_1 \cdot R_3$;
11 $Y_2 \leftarrow -Y_2 - Y_1$;
12 $R_4 \leftarrow Y_2^2$;
13 $R_4 \leftarrow R_4 - X_1 - X_2$             /* $X_{P_1-P_2}$ */;
14 $X_2 \leftarrow X_2 - X_1$;
15 $R_1 \leftarrow Y_1 \cdot X_2$               /* $Y_{\tilde{P}_1}$ */;
16 $X_2 \leftarrow R_3 - R_1$                   /* $Y_{P_1+P_2}$ */;
17 $Y_1 \leftarrow X_1 - R_4$;
18 $Y_1 \leftarrow Y_1 \cdot Y_2$;
19 $Y_2 \leftarrow Y_1 - R_1$                   /* $Y_{P_1-P_2}$ */;
20 return $\tilde{P}_1 = (X_1, R_1, Z), P_1 + P_2 = (R_2, X_2, Z), P_1 - P_2 = (R_4, Y_2, Z)$
```

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 6: `FullMult`

```
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      $Q \leftarrow$ NewAddSub($P_1, P_2$);
5      $P_1 \leftarrow Q\,[1]$                 /* $P_1 \leftarrow (P_1 + P_2)$ */;
6      $P_2 \leftarrow Q\,[2]$                 /* $P_2 \leftarrow (P_1 - P_2) = P$ */;
7      $Q \leftarrow$ NewAddSub($P_1, (-1)^{\bar{k}_i} P_2$);
8      $P_1 \leftarrow Q\,[k_i]$               /* $P_1 \leftarrow \tilde{P}_1$ or $P_1 \leftarrow P_1 + P_2$ */;
9      $P_2 \leftarrow Q\,[\bar{k}_i]$         /* $P_2 \leftarrow \tilde{P}_1$ or $P_2 \leftarrow P_1 + P_2$ */;
10 return $P_2$
```

This basic `FullMult` only uses the `NewAddSub` algorithm, for a n-bit scalar the complexity is $n(12M + 6S)$. We note that the second `NewAddSub` (Algorithm 6 line 7) is only a simple `NewAdd`. If one has enough space to code these two algorithms, a modified `FullMult'` can run in:

$$n(\texttt{NewAddSub} + \texttt{NewAdd}) = n((6M + 3S) + (5M + 2S)) = n(11M + 5S).$$

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 in the `NewAddSub` or in the `NewAdd` for computing either the X or Y coordinates. We can then reduce our `FullMult` algorithm into a `LightMult` version where we don't take care of the Z inside the loop but compute the final Z in the last round for minimal computational cost. We easily modify our `NewAddSub` into a `LightAddSub` such that

$$\texttt{LightAddSub}(P_1, P_2) \rightarrow (\tilde{P_1}, P_1 + P_2, P_1 - P_2) \text{ with } Z_{\tilde{P_1}} = Z_{P_1+P_2} = Z_{P_1-P_2},$$

where `LightAddSub` is the same algorithm as `NewAddSub` but without computing the Z. Then `LightAddSub` costs $5M + 3S$. The multiplication algorithm has to be slightly modified by computing the last round of the loop on k_i separately in order to get the right Z-coordinate. We call this algorithm `LightMult` (Algorithm 7).

If one has enough space, we can use the same trick as in `FullMult` algorithm replacing the `LightAddSub` in Algorithm 7, lines 8 and 20, with a version of the original `NewAdd` without computing the Z-coordinate called `LightAdd`. We finally obtain a modified `LightMult'` that runs in:

$$n(\texttt{LightAddSub} + \texttt{LightAdd}) = n((5M + 3S) + (4M + 2S)) = n(9M + 5S).$$

5. Resistance against side-channel attacks

Side-channel attacks are based on the observation that side-channel leakage (power consumption, electromagnetic emissions, etc.) depends on the instruction being executed, or on the data being handled.

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 then behave differently and this translates to a change of side-channel information being leaked by the device. With simple power analysis-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 equalise the side-channel leakage. The standard 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 an adapted Montgomery ladder. Our four proposed algorithms each compute 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 k. Thus, simple power analysis-like attacks are defeated. The side-channel information also becomes a fixed pattern. The Montgomery ladder is secure against SPA and its security is independant of the formulas used within the ladder.

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

```
Algorithm 7: LightMult
   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  $P_{save} \leftarrow P$;
4  for $i \leftarrow n-2$ to $1$ do
5      $Q \leftarrow$ LightAddSub($P_1, P_2$);
6      $P_1 \leftarrow Q[1]$                /* $P_1 \leftarrow (P_1 + P_2)$ */;
7      $P_2 \leftarrow Q[2]$                /* $P_2 \leftarrow (P_1 - P_2) = P$ */;
8      $Q \leftarrow$ LightAddSub($P_1, (-1)^{\bar{k}_i} P_2$);
9      $P_1 \leftarrow Q[k_i]$              /* $P_1 \leftarrow \tilde{P_1}$ or $P_1 \leftarrow P_1 + P_2$ */;
10     $P_2 \leftarrow Q[\bar{k}_i]$        /* $P_2 \leftarrow \tilde{P_1}$ or $P_2 \leftarrow P_1 + P_2$ */;
   // Last round
11 $Q \leftarrow$ LightAddSub($P_1, P_2$);
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 $Z_{final} \leftarrow X_{P_2} * Y_{P_{save}}$;
15 $Z_{final} \leftarrow (Z_{final})^{-1}$;
16 $Z_{final} \leftarrow Z_{final} * Y_{P_2}$;
17 $Z_{final} \leftarrow Z_{final} * X_{P_{save}}$;
18 $Z_{final} \leftarrow Z_{final} * Z_{P_{save}}$;
19 $Z_{final} \leftarrow (Z_{final} * (X_{P_2} - X_{P_1}))$;
20 $Q \leftarrow$ LightAddSub($P_1, (-1)^{\bar{k}_i} P_2$);
21 $P_1 \leftarrow Q[k_i]$                  /* $P_1 \leftarrow \tilde{P_1}$ or $P_1 \leftarrow P_1 + P_2$ */;
22 $P_2 \leftarrow Q[\bar{k}_i]$            /* $P_2 \leftarrow \tilde{P_1}$ or $P_2 \leftarrow P_1 + P_2$ */;
23 $P_2 \leftarrow [X_{P_2}, Y_{P_2}, Z_{final}]$;
24 return $P_2$
```

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 intial 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 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 knowns 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. Joye and Yen [**JY02**] proposed a slight modification to the Montgomery ladder in order to make it resistant to M safe-error attacks, an attack that implies stronger assumptions in the attacker's capabilities. 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.

In order to thwart many attacks, a good set of countermeasures would be: random splitting of the scalar [**CJ01**] and point verification [**BMM00**] that checks if a point lies on a curve or not. Our proposed algorithms combined with this set of countermeasures are resistant to known attacks.

6. Conclusion

We presented in this paper a new scalar multiplication algorithm for elliptic curves which is as resistant as the Montgomery ladder and faster than its adaptation for generic curves. Table 1 compares the efficiency of our algorithms with the generic Montgomery ladder algorithms. We can attain a complexity of $9M + 5S$ per bit of scalar with our `LightMult'` algorithm on any elliptic curve over a prime field

TABLE 1. Summary of scalar multiplication algorithms

	Complexity (per bit of scalar)
Generic Montgomery ladder [**BJ02**]	$15M + 5S$
Improved Izu-Takagi [**IT02**]	$13M + 4S$
FullMult	$12M + 6S$
FullMult'	$11M + 5S$
LightMult	$10M + 6S$
LightMult'	$9M + 5S$

whereas, Izu-Takagi's generic Montgomery ladder costs $13M + 4S$. We have also shown the side-channel resistance of Montgomery, type algorithms against simple side-channel attacks and fault attacks. Hence, combining one of our algorithm propositions with a DPA randomization technique will provide an efficient scalar multiplication resistant against main side-channel threats.

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 (2004), 247–256.

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

[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** (2000), 131–146.

[BOS06] J. Blömer, M. Otto, and J.P. Seifert, *Sign change fault attacks on elliptic curve cryptosystems*, FDTC 2005, LNCS **4236** (2006), 36–52.

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

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

[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** (2004), 760–768.

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

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

[GLS09] S.D. Galbraith, X. Lin, and M. Scott, *Endomorphisms for faster elliptic curve cryptography on a large class of curves*, EUROCRYPT 2009, LNCS **5479** (2009), 518–535.

[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** (2002), 371–374.

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

[JY00] M. Joye and S.M. Yen, *Optimal left-to-right binary signed-digit recoding*, IEEE Transactions on Computers **49** (2000), 740–748.

[JY02] ———, *The montgomery powering ladder*, CHES 2002, LNCS **2523** (2002), 1–11.

[LG09] P. Longa and C. Gebotys, *Fast multibase methods and other several optimizations for elliptic curve scalar multiplication*, PKC 2009, LNCS **5443** (2009), 443–462.

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

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

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

[OT04] K. Okeya and T. Takagi, *Sca-resistant and fast elliptic scalar multiplication based on wnaf*, IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences **87** (2004), 75–84.

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

[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.

IML - ERISCS, Université de la Méditerranée, Case 907, 163 Avenue de Luminy, 13288 Marseille Cedex 09, FRANCE
E-mail address: alexandre.venelli@atmel.com

ATMEL Secure Microcontroller Solutions, Zone Industrielle, 13106 Rousset, FRANCE
E-mail address: francois.dassance@atmel.com