Protections against Differential Analysis for Elliptic Curve Cryptography
– An Algebraic Approach –

Marc Joye[1] and Christophe Tymen[2]

[1] Gemplus Card International, Card Security Group
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos, France
marc.joye@gemplus.com – http://www.geocities.com/MarcJoye/
[2] École Normale Supérieure
45 rue d'Ulm, 75230 Paris, France
christophe.tymen@gemplus.com

Abstract. We propose several new methods to protect the scalar multiplication on an elliptic curve against Differential Analysis. The basic idea consists in transforming the curve through various random morphisms to provide a non-deterministic execution of the algorithm.
The solutions we suggest complement and improve the state-of-the-art, but also provide a practical toolbox of efficient countermeasures. These should suit most of the needs for protecting implementations of crypto-algorithms based on elliptic curves.

Keywords. Public-key cryptography, Side-channel attacks, Differential power analysis (DPA), Timing attacks, Elliptic curves, Smart-cards.

1 Introduction

Since the introduction of the timing attacks [10] by Paul Kocher in 1996 and subsequently of the Differential Power Analysis (DPA) [9], the so-called side-channel attacks have become a major threat against tamper-resistant devices like smart-cards, to the point where the immediate relevance of classical security notions is somewhat questionable. Furthermore, numerous experiments show that most of the time, perfunctory countermeasures do not suffice to thwart those attacks.

In the case of public-key cryptosystems based on the discrete logarithm on elliptic curves, the running time does not really represent a bottleneck for smart-card applications, which are equipped with additional devices for fast computation in finite fields. Therefore, investigating the security of these applications, less constrained by performance criteria, against side-channel attacks is very relevant.

Compared to the previous works of [5] and [7], this paper systematically develops the same idea: assuming that an elliptic curve cryptosystem executes some operations in the group of a curve E, the whole algorithm is transposed

Ç.K. Koç, D. Naccache, and C. Paar (Eds.): CHES 2001, LNCS 2162, pp. 377–390, 2001.

to a curve $\phi(E)$, where ϕ is a random morphism. The rich algebraic structure of elliptic curves enables numerous possible choices for such morphisms.

The rest of this paper is organized as follows. In the next section, we provide a brief description of elliptic curves. We refer the reader to Appendix A for further mathematical details. The general principles of differential analysis and how this can reveal the secret keys of an elliptic curve cryptosystem are explained in Section 3. Next, we provide two main classes of possible morphisms to randomize the basepoint. Finally, we present in Section 5 a new randomization of the encoding of the multiplier in the case of Anomalous Binary Curves (ABC).

2 Elliptic Curves

Let $\mathbb{K}$ be a field. An *elliptic curve over* $\mathbb{K}$ is a pair $(E, \boldsymbol{O})$ where E is a non-singular curve of genus one over $\mathbb{K}$ with a point $\boldsymbol{O} \in E$. It is well known that the set of points $(x, y) \in \mathbb{K} \times \mathbb{K}$ verifying the (non-singular) Weierstraß equation

$$E_{/\mathbb{K}} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (a_i \in \mathbb{K}) \tag{1}$$

together with $\boldsymbol{O}$ form an elliptic curve and that an elliptic curve can always be expressed in such a form. The point $\boldsymbol{O}$ is called the *point at infinity*.

The set of points (x, y) satisfying Eq. (1) and $\boldsymbol{O}$ form an Abelian group where $\boldsymbol{O}$ is the neutral element. This group is denoted by $E(\mathbb{K})$ and the group operation is denoted by $+$. The operation consisting in computing the multiple of a point, $\boldsymbol{Q} = k\boldsymbol{P} := \boldsymbol{P} + \cdots + \boldsymbol{P}$ (k times), is called *(elliptic curve) scalar multiplication*. We refer the unfamiliar reader to Appendix A for the required background on this particular topic.

3 Differential Analysis

In his CRYPTO '96 paper [10] and thereafter in [9] with Jaffe and Jun, Kocher launched a new class of attacks, the so-called *side-channel attacks*.

The basic idea of the side-channel attacks is that some side-channel information (e.g., timing, power consumption, electromagnetic radiation) of a device depends on the operations it performs. For instance, it is well known that the modification of a memory state yields a different power consumption according to the memory goes from one to zero, or the opposite.

By capturing this information, it may be possible to recover some secret keys involved during the execution of a crypto-algorithm, at least in a careless implementation. When a single input is used in eliciting information, the process is referred to as a *Simple Analysis* and when there are several inputs used together with statistical tools, it is referred to as a *Differential Analysis*. In this paper, we are concerned with the second type of attack, and in particular in the context of elliptic curve cryptography.

For elliptic curve cryptosystems, this type of attack applies to the scalar multiplication. Following [5], a simple countermeasure to defeat simple analysis attacks resides in replacing the standard double-and-add algorithm by a

double-and-add-*always* algorithm for computing $\boldsymbol{Q} = k\boldsymbol{P}$ on an elliptic curve (see also [7] for further countermeasures dedicated to ABC curves). However, such an algorithm is still susceptible to a differential analysis attack. Let $k = (k_{m-1}, \ldots, k_0)_2$ be the binary expansion of multiplier k. Suppose that an attacker already knows the highest bits, $k_{m-1}, \ldots, k_{j+1}$, of k. Then, he *guesses* that the next bit k_j is equal to one. He randomly chooses several points $\boldsymbol{P_1}, \ldots, \boldsymbol{P_t}$ and computes $\boldsymbol{Q_r} = (\sum_{i=j}^{m-1} k_i)\boldsymbol{P_r}$ for $1 \leq r \leq t$. Using a boolean selection function g, he prepares two sets: the first set, $\mathcal{S}_{\texttt{true}}$, contains the points $\boldsymbol{P_r}$ such that $g(\boldsymbol{Q_r}) = \texttt{true}$ and the second set, $\mathcal{S}_{\texttt{false}}$, contains those such that $g(\boldsymbol{Q_r}) = \texttt{false}$. Depending on the side-channel information monitored by the attacker and the actual implementation, a selection function may, for example, be the value of a given bit in the representation of $\boldsymbol{Q_r}$.

Let $C(r)$ denote the side-channel information associated to the computation of $k\boldsymbol{P_r}$ by the cryptographic device (e.g., the power consumption). If the guess $k_j = 1$ is incorrect then the difference

$$\langle C(r) \rangle_{\substack{1 \leq r \leq t \\ \boldsymbol{P_r} \in \mathcal{S}_{\texttt{true}}}} - \langle C(r) \rangle_{\substack{1 \leq r \leq t \\ \boldsymbol{P_r} \in \mathcal{S}_{\texttt{false}}}}$$

will be ≈ 0 as the two sets appear as two random (i.e. uncorrelated) sets; otherwise the guess is correct. Once k_j is known, the remaining bits, $k_{j-1}, \ldots, k_0$, are recovered recursively, in the same way. We note that such attacks are not restricted to binary methods and can be adapted to work with other scalar multiplication methods, as well.

To thwart differential attacks, it is recommended to randomize the basepoint $\boldsymbol{P}$ and the multiplier k in the computation of $\boldsymbol{Q} = k\boldsymbol{P}$. Several countermeasures are already known. See [5] for general curves and [7] for ABC curves. The next section proposes two techniques for randomizing the basepoint and Section 5 shows how to randomize the multiplier for an ABC curve.

4 Randomizing the Basepoint

4.1 Elliptic Curve Isomorphisms

We first recall some results on isomorphisms between elliptic curves. We say that two elliptic curves over a field $\mathbb{K}$ defined by their Weierstraß equations E and E' are *isomorphic over* $\mathbb{K}$ (or $\mathbb{K}$-*isomorphic*) if they are isomorphic as projective varieties. It turns out that curve isomorphisms induce group morphisms. The determination of isomorphisms between two given elliptic curves is solved in the next two corollaries.

Corollary 1. *Let* $\mathbb{K}$ *be a field with* $\operatorname{Char}\mathbb{K} \neq 2, 3$. *The elliptic curves given by* $E_{/\mathbb{K}} : y^2 = x^3 + ax + b$ *and* $E'_{/\mathbb{K}} : y^2 = x^3 + a'x + b'$ *are* $\mathbb{K}$-*isomorphic if and only if there exists* $u \in \mathbb{K}^*$ *such that* $u^4a' = a$ *and* $u^6b' = b$. *Furthermore, we have*

$$\varphi : E(\mathbb{K}) \xrightarrow{\sim} E'(\mathbb{K}), \begin{cases} \boldsymbol{O} \mapsto \boldsymbol{O} \\ (x, y) \mapsto (u^{-2}x, u^{-3}y) \end{cases} \tag{2}$$

and

$$\varphi^{-1} : E'(\mathbb{K}) \stackrel{\sim}{\to} E(\mathbb{K}), \begin{cases} \boldsymbol{O} \mapsto \boldsymbol{O} \\ (x, y) \mapsto (u^2 x, u^3 y) \end{cases} . \tag{3}$$

Proof. With the notations of Proposition 2 (in appendix), we obtain $r = s = t = 0$ and so $u^4 a'_4 = a_4$ and $u^6 a'_6 = a_6 \iff u^4 a' = a$ and $u^6 b' = b$ for some $u \in \mathbb{K}^*$. □

Corollary 2. *Let* $\mathbb{K}$ *be a field with* $\operatorname{Char} \mathbb{K} = 2$. *The (non-supersingular) elliptic curves given by* $E_{/\mathbb{K}} : y^2 + xy = x^3 + ax^2 + b$ *and* $E'_{/\mathbb{K}} : y^2 + xy = x^3 + a'x^2 + b'$ *are* $\mathbb{K}$*-isomorphic if and only if there exists* $s \in \mathbb{K}$ *such that* $a' = a + s + s^2$ *and* $b' = b$. *Furthermore, we have*

$$\varphi : E(\mathbb{K}) \stackrel{\sim}{\to} E'(\mathbb{K}), \begin{cases} \boldsymbol{O} \mapsto \boldsymbol{O} \\ (x, y) \mapsto (x, y + sx) \end{cases} \tag{4}$$

and

$$\varphi^{-1} : E'(\mathbb{K}) \stackrel{\sim}{\to} E(\mathbb{K}), \begin{cases} \boldsymbol{O} \mapsto \boldsymbol{O} \\ (x, y) \mapsto (x, y + sx) \end{cases} . \tag{5}$$

Proof. From Proposition 2, the relation $ua'_1 = a_1 + 2s$ gives $u = 1$. The third and fourth relations give $r = 0$ and $t = 0$, respectively. Hence, from the second relation we have $a'_2 = a_2 - s - s^2$ whereas the last one yields $a'_6 = a_6 \iff a' = a + s + s^2$ and $b' = b$. □

We can thus randomize the scalar multiplication algorithm as follows. We perform the scalar multiplication on a random isomorphic elliptic curve and then we come back to the original elliptic curve. More formally, if φ is a random isomorphism from $E_{/\mathbb{K}}$ to $E'_{/\mathbb{K}}$, we propose to compute $\boldsymbol{Q} = k\boldsymbol{P}$ in $E(\mathbb{K})$ according to

$$\boldsymbol{Q} = \varphi^{-1}\big(k\big(\varphi(\boldsymbol{P})\big)\big)\,, \tag{6}$$

or schematically,

$$\begin{array}{ccc} \boldsymbol{P} \in E(\mathbb{K}) & \xrightarrow{\text{mult. by } k \text{ map}} & \boldsymbol{Q} = k\boldsymbol{P} \in E(\mathbb{K}) \\ \varphi \big\downarrow & & \big\uparrow \varphi^{-1} \\ \boldsymbol{P}' \in E'(\mathbb{K}) & \xrightarrow{\text{mult. by } k \text{ map}} & \boldsymbol{Q}' = k\boldsymbol{P}' \in E'(\mathbb{K}) \end{array}$$

Corollaries 1 and 2 indicate that computing the image of a point through an elliptic curve isomorphism can be done using only a few elementary field operations. This yields a very efficient means to randomize the computation of $\boldsymbol{Q} = k\boldsymbol{P}$.

Algorithm 1 (Scalar Multiplication via Random Isomorphic Elliptic Curves for Char $\mathbb{K} \neq 2, 3$).

Input: A point $\boldsymbol{P} = (x_1, y_1) \in E(\mathbb{K})$ with $E_{/\mathbb{K}} : y^2 = x^3 + ax + b$.
An integer k.
Output: The point $\boldsymbol{Q} = k\boldsymbol{P}$.

1. Randomly choose an element $u \in \mathbb{K}^*$;
2. Form the point $\boldsymbol{P'} \leftarrow (u^{-2}\, x_1, u^{-3}\, y_1)$;
3. Evaluate $a' \leftarrow u^{-4}\, a$;
4. Compute $\boldsymbol{Q'} \leftarrow k\boldsymbol{P'}$ in $E'(\mathbb{K})$ with $E'_{/\mathbb{K}} : y^2 = x^3 + a'x + b'$;[1]
5. If $(\boldsymbol{Q'} = \boldsymbol{O})$ then return $\boldsymbol{Q} = \boldsymbol{O}$ and stop. Otherwise set $\boldsymbol{Q'} \leftarrow (x'_3, y'_3)$;
6. Return $\boldsymbol{Q} = (u^2\, x'_3, u^3\, y'_3)$.

In [5, § 5.3], Coron suggests the randomization of projective coordinates in order to blind the basepoint $\boldsymbol{P}$: $\boldsymbol{P} = (x_1, y_1)$ is randomized into $(t^2 x_1 : t^3 y_1 : t)$ in Jacobian coordinates (or into $(tx_1 : ty_1 : t)$ in homogeneous coordinates) for some $t \in \mathbb{K}^*$. The advantage of the proposed countermeasure is that, in Step 2 of Algorithm 1, we can represent $\boldsymbol{P'}$ as the projective point $\boldsymbol{P'} = (u^{-2}x_1 : u^{-3}y_1 : 1)$, that is, a point with its Z-coordinate equal to 1. This results in a faster scalar multiplication algorithm. Using the values of Table 2, we precisely quantify the number of (field) multiplications needed to compute $\boldsymbol{Q} = k\boldsymbol{P}$, considering in each case the faster coordinate system. This is summarized in the next table.[2]

Table 1. Average number of (field) multiplications to compute $\boldsymbol{Q} = k\boldsymbol{P}$.

	Random. proj. coord. ([5])				Algorithm 1	
	$a \neq -3$		$a = -3$			
Double-and-add	$17\frac{1}{2} \cdot \lvert k\rvert_2$	$(\mathcal{J}^m)$	$16 \cdot \lvert k\rvert_2$	$(\mathcal{J})$	$15 \cdot \lvert k\rvert_2$	$(\mathcal{J}^m)$
Double-and-add-or-sub.	$15\frac{1}{3} \cdot \lvert k\rvert_2$	$(\mathcal{J}^m)$	$13\frac{1}{3} \cdot \lvert k\rvert_2$	$(\mathcal{J})$	$12\frac{2}{3} \cdot \lvert k\rvert_2$	$(\mathcal{J}^m)$
Double-and-add-always	$25 \cdot \lvert k\rvert_2$	$(\mathcal{J}^c)$	$23 \cdot \lvert k\rvert_2$	$(\mathcal{J}^c)$	$21 \cdot \lvert k\rvert_2$	$(\mathcal{J})$

For fields of characteristic 2, random isomorphisms of elliptic curves cannot be considered *alone* as a means to protect against differential analysis. The x-coordinate of basepoint $\boldsymbol{P}$ remains invariant through isomorphism φ (cf. Eq. (4)) and so the resulting implementation may still be subject to a differential analysis attack. However, it can be combined with other countermeasures to offer an additional security level.

The next section presents a countermeasure that randomizes both the x- and the y-coordinates of point $\boldsymbol{P}$, whatever the characteristic of the field we are working with.

[1] Note that parameter b' is not required by the scalar multiplication algorithm.
[2] $\mathcal{J}$, $\mathcal{J}^c$ and $\mathcal{J}^m$ respectively refer to the Jacobian coordinates, Chudnovsky Jacobian coordinates and the modified Jacobian coordinates (see Appendix A.1).

4.2 Field Isomorphisms

Up to isomorphism, there is one and only one finite field L of characteristic p with p^n elements. Every such field may be viewed as the field generated (over $\mathbb{F}_p$) by a root of an irreducible monic polynomial Π of degree n. Given $\Pi(X)$, any element of L can be represented as a polynomial in $\mathbb{F}_p[X]/(\Pi)$. If $e \in L$, we note $\vartheta_\Pi(e)$ its corresponding representation in $\mathbb{F}_p[X]/(\Pi)$. From another irreducible monic polynomial $\Pi'(Y)$ of degree n, we obtain another representation for the elements $e \in L$, $\vartheta_{\Pi'}(e) \in \mathbb{F}_p[Y]/(\Pi')$. The fields $\mathbb{K} := \mathbb{F}_p[X]/(\Pi)$ and $\mathbb{K}' := \mathbb{F}_p[Y]/(\Pi')$ being isomorphic, we let ϕ denote such an isomorphism from $\mathbb{K}$ to $\mathbb{K}'$. The map ϕ extends to $\mathbb{K} \times \mathbb{K}$ with $\phi(x, y) = (\phi(x), \phi(y))$. In particular, ϕ transforms the equation of an elliptic curve over $\mathbb{K}$ into the equation of an elliptic curve over $\mathbb{K}'$, i.e.,

$$E_{/\mathbb{K}} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

is transformed into

$$E'_{/\mathbb{K}'} : y^2 + \phi(a_1)xy + \phi(a_3)y = x^3 + \phi(a_2)x^2 + \phi(a_4)x + \phi(a_6) \ .$$

Consequently, isomorphisms between fields can be used to randomize the representation of the basepoint $\boldsymbol{P}$. To compute $\boldsymbol{Q} = k\boldsymbol{P}$, we first choose randomly a field $\mathbb{K}'$ isomorphic to $\mathbb{K}$ through isomorphism ϕ. Then, we compute $\boldsymbol{Q}$ as

$$\boldsymbol{Q} = \phi^{-1}\big(k\big(\phi(\boldsymbol{P})\big)\big) \ . \tag{7}$$

In other words, we represent $\boldsymbol{P} \in E(\mathbb{K})$ as a point $\boldsymbol{P}' \in E'(\mathbb{K}')$, next we compute $\boldsymbol{Q}' := k\boldsymbol{P}'$ in $E'(\mathbb{K}')$, and finally we go back to the original representation by representing $\boldsymbol{Q}'$ as a point $\boldsymbol{Q} \in E(\mathbb{K})$.

At first glance, it is unclear that this could lead to an countermeasure efficient in a constrained environment. Indeed, to build a field $\mathbb{K}'$ isomorphic to $\mathbb{K}$, a natural way consists in determining an irreducible monic polynomial of degree n, $\Pi' \in \mathbb{F}_p[Y]$. An isomorphism ϕ is then obtained by computing a root α of Π in $\mathbb{K}'$:

$$\phi : \mathbb{K} \xrightarrow{\sim} \mathbb{K}' : x \mapsto \sum_{i=0}^{n-1} x_i \, \alpha^i \, , \tag{8}$$

where $x = \sum_{i=0}^{n-1} x_i \, X^i \in \mathbb{K} = \mathbb{F}_p[X]/(\Pi)$. Likewise, the inverse map, from $\mathbb{K}'$ to $\mathbb{K}$, requires to find a root β of Π' in $\mathbb{K}$.

However, we can do much better when some permanent writable memory is at disposal (e.g., the EEPROM in a smart-card implementation). The general idea is, given an isomorphism $\phi : \mathbb{K} \xrightarrow{\sim} \mathbb{K}'$ stored in EEPROM, to determine from ϕ and $\mathbb{K}'$ a new field $\mathbb{K}''$ and a new isomorphism $\phi' : \mathbb{K} \xrightarrow{\sim} \mathbb{K}''$, and so on. This can be done thanks to Proposition 1, which yields a recursive method for constructing irreducible polynomials of same degree.

Proposition 1. *Let T be a polynomial permutation[3] of $\mathbb{F}_{p^n}$ and let Π be an irreducible polynomial in $\mathbb{F}_p[X]$ of degree n. Then polynomial $\Pi \circ T$ has at least one irreducible factor of degree n, say Π', in $\mathbb{F}_p[X]$.*

[3] A polynomial T with coefficients in $\mathbb{F}_p$ is a *polynomial permutation of* $\mathbb{F}_{p^n}$ if the map $x \mapsto T(x)$ permutes $\mathbb{F}_{p^n}$.

Proof. Let α be a root of Π. As Π is irreducible, the orbit of α under the action of the Frobenius is of cardinality n. T being a permutation, the image of this orbit through T^{-1} is still of cardinality n. Since T is a polynomial with coefficients in $\mathbb{F}_p$, it commutes with the Frobenius, and thus the image of the orbit of α through T^{-1} appears as the orbit of $T^{-1}(\alpha)$. Consequently, the polynomial $\prod_i (X - (T^{-1}(\alpha))^{p^i})$ is irreducible of degree n, and divides $\Pi \circ T$. □

Hence, if we choose a polynomial permutation T of small degree (e.g., 2 or 3), we compute $\Pi \circ T$ and factor it with a *specific* algorithm to find Π'. A further generalization consists in storing a family of polynomial permutations $\mathfrak{S} = \{T_i\}$ in EEPROM and to randomly choose $T \in \mathfrak{S}$ when constructing Π'.

We note Π the publicly known polynomial which defines the field $\mathbb{K} = \mathbb{F}_p[X]/(\Pi)$ used as the reference field. We assume that another polynomial $\Pi^{(1)}$ defines the field $\mathbb{K}^{(1)}$ isomorphic to $\mathbb{K}$. We also assume that two polynomials $\alpha^{(1)}, \beta^{(1)} \in \mathbb{F}_p[X]$ of degree at most n verifying Eqs. (9) have initially been stored in EEPROM. These additional data must of course be kept secret. At the j^{th} execution of the scalar multiplication algorithm, the EEPROM contains an irreducible monic polynomial $\Pi^{(j)} \in \mathbb{F}_p[X]$ of degree n, and two polynomials $\alpha^{(j)}, \beta^{(j)} \in \mathbb{F}_p[X]$ such that

$$\begin{cases} \Pi(\beta^{(j)}) & \equiv 0 \quad (\Pi^{(j)}) \\ \Pi^{(j)}(\alpha^{(j)}) & \equiv 0 \quad (\Pi) \end{cases} . \tag{9}$$

These relations simply say that $\alpha^{(j)}$ and $\beta^{(j)}$ respectively define an isomorphism $\phi^{(j)}$ and its inverse from the field $\mathbb{K}$ to the field $\mathbb{F}_p[X]/(\Pi^{(j)})$ denoted by $\mathbb{K}^{(j)}$.

We are now ready to give the algorithm. We choose randomly $T \in \mathfrak{S}$ and determine an irreducible monic polynomial $\Pi^{(j+1)}$ of degree n in $\mathbb{F}_p[X]$ that divides $\Pi^{(j)} \circ T$ with a method that will be explained later. Then we set $\beta^{(j+1)} = \beta^{(j)} \circ T \bmod \Pi^{(j+1)}$, $\alpha^{(j+1)} = T^{-1}(\alpha^{(j)}) \bmod \Pi$, and we store $\alpha^{(j+1)}, \beta^{(j+1)}$ and $\Pi^{(j+1)}$ in EEPROM. Here, T^{-1} denotes the permutation inverse of T. It is easy to check that $\alpha^{(j+1)}, \beta^{(j+1)}$ and $\Pi^{(j+1)}$ still verify Eqs. (9), and thus define an isomorphism $\phi^{(j+1)}$ and its inverse from $\mathbb{K}$ to $\mathbb{F}_p[X]/(\Pi^{(j)})$. It remains to compute $\boldsymbol{P'} = \phi^{(j+1)}(\boldsymbol{P})$ and the coefficients of E'. Finally, we compute $k\boldsymbol{P'}$ in E' and convert the resulting point by the inverse isomorphism to obtain $\boldsymbol{Q} = k\boldsymbol{P}$. From the viewpoint of the running time, one of the advantages of this method is to skip the root finding step.

We still have to show how to solve Step 2 in the above algorithm. We illustrate the technique in the case $\mathbb{K} = \mathbb{F}_2[X]/(\Pi)$ with Π of degree n and $\gcd(2^n - 1, 3) = 1$ (this case includes the popular choice $n = 163$ for elliptic curve cryptosystems), but we stress that the proposed technique is fully general and can be adapted to the other cases, as well.

First, note that:

Lemma 1. *If* $\gcd(2^n - 1, 3) = 1$ *then the elements of*

$$\mathfrak{S} = \left\{X^3, 1 + X^3, X + X^2 + X^3, 1 + X + X^2 + X^3\right\} \subset \mathbb{F}_2[X]$$

permute $\mathbb{F}_{2^n}$.

Algorithm 2 (Scalar Multiplication via Random Isomorphic Fields).

Input: A point $\boldsymbol{P} = (x_1, y_1) \in E(\mathbb{K})$
with $\begin{cases} E_{/\mathbb{K}} : y^2 + xy = x^3 + ax^2 + b & \text{if } \mathrm{Char}\,\mathbb{K} = 2 \\ E_{/\mathbb{K}} : y^2 = x^3 + ax + b & \text{if } \mathrm{Char}\,\mathbb{K} > 3 \end{cases}$.
An integer k.
[EEPROM: Polynomials $\alpha^{(j)}$, $\beta^{(j)}$ and $\Pi^{(j)}$.]
Output: The point $\boldsymbol{Q} = k\boldsymbol{P}$.

1. Randomly choose $T \in \mathfrak{S}$;
2. Determine, in $\mathbb{F}_p[X]$, an irreducible monic polynomial $\Pi^{(j+1)}$ s.t.
$$\Pi^{(j+1)} \text{ divides } \Pi^{(j)} \circ T;$$
3. Set $\beta^{(j+1)} \leftarrow \beta^{(j)} \circ T \bmod \Pi^{(j+1)}$;
4. Set $\alpha^{(j+1)} \leftarrow T^{-1}(\alpha^{(j)}) \bmod \Pi$;
5. Update the EEPROM with $\alpha^{(j+1)}$, $\beta^{(j+1)}$ and $\Pi^{(j+1)}$;
6. Form $\boldsymbol{P}' \leftarrow \boldsymbol{P} \circ \beta^{(j+1)} \bmod \Pi^{(j+1)}$;
7. Evaluate $a' \leftarrow a \circ \beta^{(j+1)} \bmod \Pi^{(j+1)}$;
8. Compute $\boldsymbol{Q}' \leftarrow k\boldsymbol{P}'$ in $E'\big(\mathbb{F}_p[X]/(\Pi^{(j+1)})\big)$;
9. If $(\boldsymbol{Q}' = \boldsymbol{O})$ then return $\boldsymbol{Q} = \boldsymbol{O}$ and stop. Otherwise set $\boldsymbol{Q}' \leftarrow (x'_3, y'_3)$;
10. Return $\boldsymbol{Q} = (x'_3, y'_3) \circ \alpha^{(j+1)} \bmod \Pi$.

Proof. Let α be a primitive element of $\mathbb{F}^*_{2^n}$ (remember that $\mathbb{F}_{2^n} = \mathbb{F}^*_{2^n} \cup \{0\}$). Then $\langle \alpha^3 \rangle$ generates a subgroup of order $(2^n - 1)/\gcd(2^n - 1, 3) = 2^n - 1$ and so α^3 is a primitive element or equivalently X^3 permutes $\mathbb{F}_{2^n}$. Suppose that there exist $\alpha, \beta \in \mathbb{F}_{2^n}$ s.t. $\alpha^3 + 1 = \beta^3 + 1 \iff \alpha^3 = \beta^3$. This implies $\alpha = \beta$ since X^3 is a permutation polynomial. The remaining cases are proved similarly by noting that $\alpha^2 - \beta^2 = (\alpha - \beta)^2$. □

Given a set $\mathfrak{S}$ of permutation polynomials, write $Q := \Pi^{(j)} \circ T$ for some $T \in \mathfrak{S}$ and Π irreducible of degree n. The fact that Q has degree $3n$ enables us to specialize the classical factorization algorithms (see, e.g., [3, p. 125]):

1. Compute $R = X^{2^n} - X \bmod Q$;
2. Then, using Proposition 1, $\Pi' = \gcd(Q, R)$ is irreducible of degree n in $\mathbb{F}_2[X]$.

5 Randomizing the Multiplier on ABC Curves

The other side of countermeasures for elliptic curve cryptography is the introduction of a random to blind the multiplier during the scalar multiplication. This technique is useful to prevent Differential Analysis, but may also contribute to an additional security level against Simple Analysis, as the multiplier is in general secret.

The proposed method is specific to ABC curves (see Appendix A.2 for the definitions) where the multiplier first goes through several encoding functions before being used in the scalar multiplication loop itself. We take advantage of the properties of this encoding to randomize the multiplier.

Building on previous works by Koblitz [8] and Meier-Staffelbach [11], Solinas presents in [14] a very efficient algorithm to compute $\boldsymbol{Q} = k\boldsymbol{P}$ on an ABC curve. Letting $\tau : (x, y) \mapsto (x^2, y^2)$ the Frobenius endomorphism, his algorithm proceeds as follows.

1. Compute, in $\mathbb{Z}[\tau]$, $\kappa \leftarrow k \bmod (\tau^n - 1)$;
2. Using [14, Algorithm 4], evaluate the τ-NAF of κ, $\kappa = \sum_i k_i \tau^i$;
3. Compute $\boldsymbol{Q} \leftarrow k\boldsymbol{P}$ as $\boldsymbol{Q} = \sum_i k_i \tau^i(\boldsymbol{P})$;
4. Return $\boldsymbol{Q}$.

Our randomization method exploits the structure of $\mathbb{Z}[\tau] \subseteq \mathrm{End}(E)$. Let $\rho \in \mathbb{Z}[\tau]$. If $x \equiv y \pmod{\rho(\tau^n - 1)}$ then x and y still act identically on the curve. Consequently, instead of reducing the multiplier modulo $\tau^n - 1$ (cf. Step 1 in the previous algorithm), we can reduce it modulo $\rho(\tau^n - 1)$ where ρ is a random element of $\mathbb{Z}[\tau]$. The length of the τ-NAF produced is approximately equal to $n + \log_2 \mathrm{N}(\rho)$, which penalizes the scalar multiplication by $\log_2 \mathrm{N}(\rho)$ additional steps. This enables to control very easily the trade-off between the running time and the expected security. Typically, for $n = 163$, we might impose that $\mathrm{N}(\rho) \approx 2^{40}$, which roughly produces τ-NAF of 200 digits (in $\{-1, 0, 1\}$) instead of 160 with the deterministic method. The detailed algorithm is presented below.

Algorithm 3 (Scalar Multiplication via Random Exponent Recoding for ABC Curves).

Input: A point $\boldsymbol{P} = (x_1, y_1) \in E(\mathbb{F}_{2^n})$, an ABC curve.
An integer k.
A trade-off parameter l (typically, $l = 40$).
Output: The point $\boldsymbol{Q} = k\boldsymbol{P}$.

1. Randomly choose an element $\rho \in \mathbb{Z}[\tau]$ with $\mathrm{N}(\rho) < 2^l$;
2. Compute, in $\mathbb{Z}[\tau]$, $\kappa' \leftarrow k \bmod \rho(\tau^n - 1)$;
3. Evaluate the τ-NAF of κ', $\kappa' = \sum_i \kappa'_i \tau^i$;
4. Compute $\boldsymbol{Q} \leftarrow \sum_i \kappa'_i \tau^i(\boldsymbol{P})$;
5. Return $\boldsymbol{Q}$.

An interesting feature of this algorithm is that no additional routine needs to be implemented. It only requires a slight modification of the deterministic version. Furthermore, the random component ρ is spread over the full length of the multiplier. This may be better than simply adding to the multiplier a multiple of the order of the curve, as was suggested in [5, §. 5.1].

6 Conclusion

We proposed two new methods to blind the basepoint for an elliptic curve cryptosystem. These methods come from the idea of transposing the computation in another curve through a random morphism. In addition, we presented a new technique to randomize the encoding of the multiplier in the case of anomalous binary curves.

References

1. IEEE Std 1363-2000. *IEEE Standard Specifications for Public-Key Cryptography.* IEEE Computer Society, August 29, 2000.
2. D.V. Chudnovsky and G.V. Chudnovsky. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Advances in Applied Math.*, 7:385–434, 1986/7.
3. Henri Cohen. *A Course in Computational Algebraic Number Theory.* Number 138 in Graduate Texts in Mathematics. Springer-Verlag, 1993.
4. Henri Cohen, Atsuko Miyaji, and Takatoshi Ono. Efficient elliptic curve exponentiation using mixed coordinates. In K. Ohta and D. Pei, editors, *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 51–65. Springer-Verlag, 1998.
5. Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES '99)*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer-Verlag, 1999.
6. Daniel M. Gordon. A survey on fast exponentiation methods. *Journal of Algorithms*, 27:129–146, 1998.
7. M. Anwar Hasan. Power analysis attacks and algorithmic approaches to their countermeasures for Koblitz curve cryptosystems. In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 93–108. Springer-Verlag, 2000.
8. Neal Koblitz. CM-curves with good cryptographic protocols. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 279–287. Springer-Verlag, 1992.
9. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.
10. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.
11. W. Meier and O. Staffelbach. Efficient multiplication on certain non-supersingular elliptic curves. In E.F. Brickell, editor, *Advances in Cryptology – CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 333–344. Springer-Verlag, 1993.
12. Alfred J. Menezes. *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, 1993.
13. Atsuko Miyaji, Takatoshi Ono, and Henri Cohen. Efficient elliptic curve exponentiation. In Y. Han, T. Okamoto, and S. Qing, editors, *Information and Communications Security (ICICS '97)*, volume 1334 of *Lecture Notes in Computer Science*, pages 282–290. Springer-Verlag, 1997.
14. Jerome A. Solinas. An improved algorithm for arithmetic on a family of elliptic curves. In B. Kaliski, editor, *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 357–371. Springer-Verlag, 1997.

A Mathematical Background

This appendix details the elliptic curve addition formulæ. It also reviews some well-known techniques for computing $\boldsymbol{Q} = k\boldsymbol{P}$ in an elliptic curve $E(\mathbb{K})$. An

excellent survey by Gordon, including the most recent developments, can be found in [6].

In the sequel, we only consider the cases of a field $\mathbb{K}$ with $\mathrm{Char}\,\mathbb{K} \neq 2, 3$ and $\mathrm{Char}\,\mathbb{K} = 2$. In these cases, the general Weierstraß equation (cf. Eq. (1)) can be simplified considerably through an appropriate *admissible change of variables*. This is explicited in the next proposition.

Proposition 2 ([12, Theorem 2.2]). *The elliptic curves given by the Weierstraß equations*

$$E_{/\mathbb{K}} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \text{ and}$$
$$E'_{/\mathbb{K}} : y^2 + a'_1xy + a'_3y = x^3 + a'_2x^2 + a'_4x + a'_6$$

are isomorphic over $\mathbb{K}$ if and only if there exists $u \in \mathbb{K}^*$ and $r, s, t \in \mathbb{K}$ such that the change of variables

$$(x, y) \leftarrow (u^2x + r, u^3y + u^2sx + t)$$

transforms equation E into equation E'. Such a transformation is referred to as an admissible change of variables. Furthermore,

$$\begin{cases} ua'_1 = a_1 + 2s\,, \\ u^2a'_2 = a_2 - sa_1 + 3r - s^2\,, \\ u^3a'_3 = a_3 + ra_1 + 2t\,, \\ u^4a'_4 = a_4 - sa_3 + 2ra_2 - (t + rs)a_1 + 3r^2 - 2st\,, \\ u^6a'_6 = a_6 + ra_4 - ta_3 + r^2a_2 - rta_1 + r^3 - t^2\,. \end{cases}$$

A.1 Elliptic Curves over a Field $\mathbb{K}$ with $\mathrm{Char}\,\mathbb{K} \neq 2, 3$

When the characteristic of field $\mathbb{K}$ is different from $2, 3$, the Weierstraß equation of an elliptic curve can be simplified to:

$$E_{/\mathbb{K}} : y^2 = x^3 + ax + b \qquad (4a^3 + 27b^2 \neq 0)\ . \tag{10}$$

For any $\boldsymbol{P} \in E(\mathbb{K})$, we have $\boldsymbol{P} + \boldsymbol{O} = \boldsymbol{O} + \boldsymbol{P} = \boldsymbol{P}$. Let $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_2, y_2) \in E(\mathbb{K})$. The inverse of $\boldsymbol{P}$ is $-\boldsymbol{P} = (x_1, -y_1)$. If $\boldsymbol{Q} = -\boldsymbol{P}$ then $\boldsymbol{P} + \boldsymbol{Q} = \boldsymbol{O}$; otherwise the sum $\boldsymbol{P} + \boldsymbol{Q} = (x_3, y_3)$ is given by

$$x_3 = \lambda^2 - x_1 - x_2\,, \quad y_3 = \lambda(x_1 - x_3) - y_1 \tag{11}$$

with $\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}\,, & \text{if } \boldsymbol{P} \neq \boldsymbol{Q}\,, \\ \dfrac{3x_1^2 + a}{2y_1}\,, & \text{if } \boldsymbol{P} = \boldsymbol{Q}\,. \end{cases}$

To avoid the division in the computation of λ, one usually works in projective coordinates. There are basically two ways to project Eq. (10): (i) set $x = X/Z$ and $y = Y/Z$, that is, $(X : Y : Z)$ are the *homogeneous coordinates*; or (ii) set $x = X/Z^2$ and $y = Y/Z^3$, $(X : Y : Z)$ are then referred to as the *Jacobian coordinates*. Hence, to compute $\boldsymbol{Q} = k\boldsymbol{P}$ on an elliptic curve, one first represents point $\boldsymbol{P} = (x_1, y_1)$ as $(X_1 : Y_1 : Z_1)$, computes $(X_2 : Y_2 : Z_2) = k(X_1 : Y_1 : Z_1)$, and recovers $\boldsymbol{Q} = (x_2, y_2)$ from its projective form.

Homogeneous coordinates. In homogeneous coordinates, $(X : Y : Z)$ and $(tX : tY : tZ)$ (with $t \in \mathbb{K}^*$) are two equivalent representations of a same point. The point at infinity $\boldsymbol{O}$ is represented by $(0 : 1 : 0)$; it is the only point with its Z-coordinate equal to 0. Putting $x = X/Z$ and $y = Y/Z$ in Eq. (12), the Weierstraß equation of an elliptic curve becomes

$$E_{/\mathbb{K}} : Y^2Z = X^3 + aXZ^2 + bZ^3 \ . \tag{12}$$

The formula to double a point $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ is $2\boldsymbol{P} = (X_3 : Y_3 : Z_3)$, where

$$X_3 = SH, \quad Y_3 = W(T - H) - 2M^2 \quad \text{and} \quad Z_3 = S^3 \tag{13}$$

with $W = 3X_1^2 + aZ_1^2$, $S = 2Y_1Z_1$, $M = Y_1S$, $T = 2X_1M$ and $H = W^2 - 2T$. This requires 12 multiplications. Notice that if $a = -3$ then $W = 3(X_1 - Z_1)(X_1 + Z_1)$; in that case, the number of multiplications decreases to 10. The sum $\boldsymbol{R} = (X_3 : Y_3 : Z_3)$ of two points $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ and $\boldsymbol{Q} = (X_2 : Y_2 : Z_2)$ (with $\boldsymbol{P} \neq \pm\boldsymbol{Q}$) is given by

$$X_3 = WX_3', \quad 2Y_3 = RV - MW^3 \quad \text{and} \quad Z_3 = Z'W^3 \tag{14}$$

with $U_1 = X_1Z_2$, $U_2 = X_2Z_1$, $S_1 = Y_1Z_2$, $S_2 = Y_2Z_1$, $T = U_1 + U_2$, $W = U_1 - U_2$, $M = S_1 + S_2$, $R = S_1 - S_2$, $Z' = Z_1Z_2$, $H = TW^2$, $X_3' = -H + Z'R^2$ and $V = H - 2X_3'$. The addition of two points can thus be done with only 14 multiplications. If one of the two points has its Z-coordinate equal to 1 then the number of multiplications decreases to 11.

Jacobian coordinates. The use of Jacobian coordinates is suggested in the P1363 IEEE Standard [1] because it allows faster arithmetic [2]. In Jacobian coordinates, also, the representation of points is not unique, $(X : Y : Z)$ and $(t^2X : t^3Y : tZ)$ (with $t \in \mathbb{K}^*$) are equivalent representations. The Weierstraß equation is given by

$$E_{/\mathbb{K}} : Y^2 = X^3 + aXZ^4 + bZ^6 \tag{15}$$

and the point at infinity is represented by $(1 : 1 : 0)$.

The double of point $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ is equal to $2\boldsymbol{P} = (X_3 : Y_3 : Z_3)$ where

$$X_3 = M^2 - 2S, \quad Y_3 = M(S - X_3) - T \quad \text{and} \quad Z_3 = 2Y_1Z_1 \tag{16}$$

with $M = 3X_1^2 + aZ_1^4$, $S = 4X_1Y_1^2$ and $T = 8Y_1^4$. So doubling a point requires 10 multiplications. Here too, we see that the value $a = -3$ enables to reduce the number of multiplications; in this case, it decreases to 8.

The sum $\boldsymbol{R} = (X_3 : Y_3 : Z_3)$ of points $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ and $\boldsymbol{Q} = (X_2 : Y_2 : Z_2)$ (with $\boldsymbol{P} \neq \pm\boldsymbol{Q}$) is given by

$$X_3 = R^2 - TW^2, \quad 2Y_3 = RV - MW^3 \quad \text{and} \quad Z_3 = Z_1Z_2W \tag{17}$$

Table 2. Number of multiplications in addition formulæ.

	Addition		Doubling	
	$Z_2 \neq 1$	$Z_2 = 1$	$a \neq -3$	$a = -3$
Homogeneous coord.	14	11	12	10
Jacobian coord.	16	11	10	8
Chudnovsky Jacobian coord.	14	11	11	9
Modified Jacobian coord.	19	14	8	8

with $U_1 = X_1 Z_2^2$, $U_2 = X_2 Z_1^2$, $S_1 = Y_1 Z_2^3$, $S_2 = Y_2 Z_1^3$, $T = U_1 + U_2$, $W = U_1 - U_2$, $M = S_1 + S_2$, $R = S_1 - S_2$ and $V = TW^2 - 2X_3$. An addition requires 16 multiplications. When one of the two points has its Z-coordinate equal to 1 then an addition requires only 11 multiplications. A slightly different (but equally efficient) formula for addition can be found in [13]. Using the same notations as above, the sum $\boldsymbol{R} = (X_3 : Y_3 : Z_3)$ is then given by

$$X_3 = R^2 - TW^2, \quad Y_3 = -RX_3 + (RU_1 - S_1)W^2 \quad \text{and} \quad Z_3 = Z_1 Z_2 W \ . \ (17')$$

Mixed coordinates. In the general case, we have seen that Jacobian coordinates offer a faster doubling but a slower addition than homogeneous coordinates (see Table 2). Chudnovsky and Chudnovsky [2] proposed to internally represent a point $(X : Y : Z)$ in Jacobian coordinates as a 5-tuple (X, Y, Z, Z^2, Z^3). In *Chudnovsky Jacobian coordinates*, the addition formula for $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ and $\boldsymbol{Q} = (X_2 : Y_2 : Z_2)$, respectively represented as $(X_1, Y_1, Z_1, Z_1^2, Z_1^3)$ and $(X_2, Y_2, Z_2, Z_2^2, Z_2^3)$, remains the same as given by Eq. (17). The advantage is that the values of Z_1^2, Z_1^3, Z_2^2 and Z_2^3 being available, they do not have to be computed; only Z_3^2 and Z_3^3 have to be be computed to represent the result $\boldsymbol{R} = \boldsymbol{P} + \boldsymbol{Q} = (X_3 : Y_3 : Z_3)$ as the 5-tuple $(X_3, Y_3, Z_3, Z_3^2, Z_3^3)$. Therefore, Chudnovsky Jacobian coordinates require $(4 - 2) = 2$ multiplications less than ordinary Jacobian coordinates to add two points. On the other hand, the doubling is more expensive: it requires $(2-1) = 1$ multiplication more for computing $\boldsymbol{R} = 2\boldsymbol{P} = (X_3 : Y_3 : Z_3)$ since Z_1^2 has not to be computed (see Eq. (16)) but Z_3^2 and Z_3^3 have to.

The above strategy was optimized by Cohen, Miyaji and Ono [4] in order to provide the fastest known doubling algorithm on a general elliptic curve. With their coordinates, called *modified Jacobian coordinates*, a point $(X : Y : Z)$ is internally represented as a 4-tuple (X, Y, Z, aZ^4). A point is doubled with only 8 multiplications *whatever* the value of parameter a. However, this fast doubling is done at the expense of a slower addition: 19 multiplications are required to add two points in the general case and 14 multiplications when one of the two points has its Z-coordinate equal to 1.

A.2 Elliptic Curves over a Field $\mathbb{K}$ with Char $\mathbb{K} = 2$

For fields of characteristic 2, the simplified Weierstraß equation depends on whether the curve is supersingular or not. For cryptographic applications, we

are only interested in non-supersingular curves. In that case, it can be shown that an admissible change of variables yields the simplified Weierstraß equation

$$E_{/\mathbb{K}} : y^2 + xy = x^3 + ax^2 + b \qquad (b \neq 0) \ . \tag{18}$$

$\boldsymbol{O}$ being the neutral element, we have $\boldsymbol{P}+\boldsymbol{O} = \boldsymbol{O}+\boldsymbol{P} = \boldsymbol{P}$ for any $\boldsymbol{P} \in E(\mathbb{K})$. Let $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_2, y_2) \in E(\mathbb{K})$. The inverse of $\boldsymbol{P}$ is $-\boldsymbol{P} = (x_1, x_1 + y_1)$. If $\boldsymbol{Q} = -\boldsymbol{P}$ then $\boldsymbol{P} + \boldsymbol{Q} = \boldsymbol{O}$; otherwise the sum $\boldsymbol{P} + \boldsymbol{Q} = (x_3, y_3)$ is calculated as follows.

– If $\boldsymbol{P} \neq \boldsymbol{Q}$ then

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \,, \quad y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \tag{19}$$

with $\lambda = \dfrac{y_1 + y_2}{x_1 + x_2}$.

– If $\boldsymbol{P} = \boldsymbol{Q}$ then

$$x_3 = \lambda^2 + \lambda + a \,, \quad y_3 = x_1^2 + (\lambda + 1)x_3 \tag{20}$$

with $\lambda = x_1 + \dfrac{y_1}{x_1}$.

An important subclass of elliptic curves has been introduced by Koblitz in [8]: the *Anomalous Binary Curves* (or ABC curves in short), sometimes also referred to as *Koblitz curves*. These are elliptic curves given by Eq. (18) with $b = 1$ and $a \in \{0, 1\}$. For such curves, the Frobenius endomorphism, $\tau : (x, y) \mapsto (x^2, y^2)$, satisfies the characteristic equation

$$u^2 - (-1)^{1-a}\, u + 2 = 0 \ .$$

Koblitz suggests to speed the computation of $\boldsymbol{Q} = k\boldsymbol{P}$ by noticing that $2\boldsymbol{P} = (-1)^{1-a}\tau(\boldsymbol{P}) - \tau^2(\boldsymbol{P})$. He also suggests to write k as a Frobenius expansion since scalar multiplication by k is an endomorphism and $\mathbb{Z} \subseteq \mathbb{Z}[\tau] \subseteq \mathrm{End}(E)$. The ring $\mathbb{Z}[\tau]$ is an Euclidean domain with respect to the norm $\mathrm{N}(r + s\tau) = r^2 + (-1)^{1-a}\, rs + 2s^2$. Furthermore, as $\mathrm{N}(\tau) = 2$, every element $r + s\tau$ in $\mathbb{Z}[\tau]$ can be written as a *τ-adic non-adjacent form* (τ-NAF, in short), that is,

$$r + s\tau = \sum_i k_i\, \tau^i \quad \text{with} \begin{cases} k_i \in \{-1, 0, 1\} \\ k_i \cdot k_{i+1} = 0 \end{cases} \ . \tag{21}$$

As already remarked in [8], the drawback in this method is that the Frobenius expansion (21) is roughly twice longer than the usual balanced binary expansion and so, even if the evaluation of τ is very fast, it is not clear that the resulting method is faster. The drawback was loopholed in [11,14] with the following observation. We obviously have $\tau^n = 1$ and thus $\boldsymbol{Q} = k'\boldsymbol{P}$ with $k' = k \bmod (\tau^n - 1)$. As $\mathrm{N}(\tau^n - 1) = \#E_a(\mathbb{F}_{2^n}) \approx 2^n$ by Hasse's Theorem, the τ-NAF expression of k', $k' = \sum_i k'_i\, \tau^i$, would have a length approximatively equal to that of the (usual) NAF expression of k. The non-adjacency property (i.e., $k'_i \cdot k'_{i+1} = 0$) implies that, on average, only one third of the digits are nonzero [6]. Together with the property that the evaluation of $\tau\boldsymbol{P}$ is very fast, this yields a very efficient algorithm for computing $\boldsymbol{Q} = k\boldsymbol{P}$.