Montgomery Scalar Multiplication for Genus 2 Curves

Sylvain Duquesne

Université Montpellier II, Laboratoire I3S, UMR CNRS 5149
Place Eugène Bataillon CC 051, 34005 Montpellier Cedex, France
duquesne@math.univ-montp2.fr

Abstract. Using powerful tools on genus 2 curves like the Kummer variety, we generalize the Montgomery method for scalar multiplication to the jacobian of these curves. Previously this method was only known for elliptic curves. We obtain an algorithm that is competitive compared to the usual methods of scalar multiplication and that has additional properties such as resistance to timings attacks. This algorithm has very important applications in cryptography using hyperelliptic curves and more particularly for people interested in cryptography on smart cards.

1 Introduction

In this paper we are dealing with the scalar multiplication on the jacobian of curves defined over a field of large characteristic. One of the motivations is that this operation is the main part in cryptography based on the jacobian of curves which is becoming more and more popular. For elliptic curves, Montgomery [24] developed a method for certain curves (which are said to be in the Montgomery form) which allows faster scalar multiplication than the usual methods of exponentiation for groups. This method has the extra advantage that it is resistant to side-channel attacks which is very interesting for people who want to use elliptic curves in cryptography on smart cards. The aim of this paper is the generalization of this method to genus 2 curves. In the following, $\mathbf{K}$ will denote a field of characteristic $k \geq 7$. For cryptographic application, the base field we have in mind is $\mathbb{F}_p$ where p is prime.

2 The Montgomery Method for Scalar Multiplication on Elliptic Curves

Let E be an elliptic curve defined over $\mathbf{K}$ by the equation

$$y^2 = x^3 + a_4 x + a_6 \ .$$

Every elliptic curve defined over $\mathbf{K}$ is isomorphic to a curve given by such an equation which is called the short Weierstrass form. The set $E(\mathbf{K})$ of the points $P = (x, y)$ verifying this equation with x and y in $\mathbf{K}$, forms (together with the point at infinity) a group which will be denoted additively. The problem we are interested in is the following :

Scalar multiplication

Given a point $P \in E(\mathbf{K})$ and an integer n, compute nP as fast as possible.

Of course there are a lot of very old methods to do this, such as the classical double and add algorithm and its variants (like the sliding window method). To improve these algorithms one can choose other systems of coordinates (i.e. other means to represent points on the curve) [4]. For example, the best-known coordinates are projective ones. They are obtained by introducing a new coordinate, usually called Z, which is the lcm of the denominators of x and y. Of course, x and y are replaced by X and Y such that $x = X/Z$ and $y = Y/Z$. This choice of coordinates allows to avoid inversions, which are very costly operations. In the following, we will work with projective coordinates for reasons of efficiency. Nevertheless, the same work can be done with other systems of coordinates. The algorithm we will present here is slightly different from the usual exponentiation algorithms in the sense that the purpose is not to minimize the number of operations but rather the cost of each operation.

2.1 The Algorithm

The original idea of Montgomery [24] was to avoid the computation of the y-coordinate, so that one can hope that basic operations (doubling and adding) are easier to compute. Since for any x-coordinate, there are two corresponding points on the curve (x, y) and $(x, -y)$, this restriction is equivalent to identifying a point on the curve and its opposite. When trying to add two points $\pm P$ and $\pm Q$, one cannot decide if the result obtained is $\pm(P + Q)$ as required or $\pm(P - Q)$. Nevertheless, some operations remain possible like doubling since it is not difficult to decide if the result is $\pm 2P$ or the point at infinity. Unfortunately, doubling is not sufficient for a complete scalar multiplication: one really needs to perform some additions. In fact additions are possible if the difference $P - Q$ is known. The principle of the computation of nP is to use pairs of consecutive powers of P, so that the difference between the two components of the pair is always known and equals to P. The algorithm for scalar multiplication is as follows:

Algorithm 1. *Montgomery scalar multiplication algorithm on elliptic curves*
Input : $P \in E(\mathbf{K})$ *and* $n \in \mathbb{Z}$.
Output : x and z-coordinate of nP.

Step 1. Initialize $Q = (Q_1, Q_2) = (\mathcal{O}, P)$ *where* $\mathcal{O}$ *is the point at infinity.*

Step 2. If the bit of n is 0, $Q = (2Q_1, Q_1 + Q_2)$.

Step 3. If the bit of n is 1, $Q = (Q_1 + Q_2, 2Q_2)$.

Step 4. After doing that for each bit of n, return Q_1.

In fact, at each step, $Q = (kP, (k + 1)P)$ for some k and we compute either $(2kP, (2k + 1)P)$ or $((2k + 1)P, (2k + 2)P)$ in the following step, so that we always have $Q_2 - Q_1 = P$.
Let us note that contrary to double and add or sliding window methods, both

an addition and a doubling are done for each bit of the exponent. It is the price to be paid to avoid the computation of the y-coordinate but we hope that the gain obtained thanks to this restriction will be sufficient to compensate for the large number of operations. That is the reason why the Montgomery form for elliptic curves has been introduced. The interested reader will find more details for this section in [24] or [26].

2.2 The Montgomery Form

An elliptic curve E is transformable into the Montgomery form if it is isomorphic to a curve given by an equation of the type

$$E_m : By^2 = x^3 + Ax^2 + x \ .$$

It is easy to prove ([26]) that E is transformable in the Montgomery form if and only if

- the polynomial $x^3 + a_4x + a_6$ has at least one root α in $\mathbf{K}$,
- the number $3\alpha^2 + a_4$ is a square in $\mathbf{K}$.

Thus all elliptic curves are not transformable into the Montgomery form. Nevertheless, since the two coefficients can be chosen arbitrarily in $\mathbf{K}$, the number of curves in such a form is of the same order as for general elliptic curves (for example $O(p^2)$ if $\mathbf{K} = \mathbb{F}_p$).
Please note that the first condition means that there is at least one 2-torsion point on the curve E, so that the cardinality of the curve is even.

2.3 Formulas for Doubling and Adding

Let us now describe the arithmetic of curves in the Montgomery form.

Proposition 1. *Let $\mathbf{K}$ be a field of characteristic $k \neq 2, 3$ and let E_m be an elliptic curve defined over $\mathbf{K}$ in the Montgomery form. Let $P = (X_p, Y_p, Z_p)$ and $Q = (X_q, Y_q, Z_q) \in E_m(\mathbf{K})$ be given in projective coordinates. Assume that the difference $P - Q = (x, y)$ is known in affine coordinates. Then we obtain the X and Z-coordinates for $P + Q$ and $2P$ by the following formulas :*

$$\begin{aligned} X_{p+q} &= ((X_p - Z_p)(X_q + Z_q) + (X_p + Z_p)(X_q - Z_q))^2 \ , \\ Z_{p+q} &= x\,((X_p - Z_p)(X_q + Z_q) - (X_p + Z_p)(X_q - Z_q))^2 \ , \end{aligned}$$

$$\begin{aligned} 4X_pZ_p &= (X_p + Z_p)^2 - (X_p - Z_p)^2 \ , \\ X_{2p} &= (X_p + Z_p)^2(X_p - Z_p)^2 \ , \\ Z_{2p} &= 4X_pZ_p\left((X_p - Z_p)^2 + \frac{A+2}{4}4X_pZ_p\right) \ . \end{aligned}$$

In this way, both an addition and a doubling take 3 multiplications and 2 squares each so that the cost of this algorithm is about $10|n|_2$ multiplications where $|n|_2$ denotes the number of bits of n.
In the best case with usual scalar multiplication, one needs 4 multiplications and 4 squaring just for doubling and more for adding. Thus, for curves in the Montgomery form, this method is interesting. In practice, the gain obtained is about 10 percent (compared in [6] for 192 bits with a sliding window method of size 4 after a Koyama-Tsuruoka recoding [18] and using mixed jacobian modified coordinates [4]).

2.4 General Case

We are now interested in a Montgomery method for scalar multiplication for elliptic curves which cannot be transformed into the Montgomery form. In fact the method for scalar multiplication is the same, we just need to have formulas for doubling and adding. These formulas can be found in [1], [10] or [13]. Let us recall them.

Proposition 2. *Let* $\mathbf{K}$ *be a field of characteristic* $k \neq 2, 3$ *and let* E *be an elliptic curve defined over* $\mathbf{K}$ *as described in Sect. 2. Let* $P = (X_p, Y_p, Z_p)$ *and* $Q = (X_q, Y_q, Z_q) \in E(\mathbf{K})$ *be given in projective coordinates. Assume that the difference* $P - Q = (x, y)$ *is known in affine coordinates. Then we obtain the* X *and* Z*-coordinates for* $P + Q$ *and* $2P$ *by the following formulas :*

$$\begin{aligned} X_{p+q} &= -4a_6 Z_p Z_q (X_p Z_q + X_q Z_p) + (X_p X_q - a_4 Z_p Z_q)^2 \ , \\ Z_{p+q} &= x(X_p Z_q - X_q Z_p) \ , \\ X_{2p} &= \left(X_p^2 - a_4 Z_p^2\right)^2 - 8a_6 X_p Z_p^3 \ , \\ Z_{2p} &= 4Z_p \left(X_p^3 + a_4 X_p Z_p^2 + a_6 Z_p^3\right) \ . \end{aligned}$$

Addition can be evaluated in 10 multiplications and doubling in 9. Thus, in this way, the scalar multiplication can be performed in about $19|n|_2$ multiplications in the base field. This method can even be optimized in most cases to $17|n|_2$ multiplications [7]. Of course, in this case, the algorithm is not interesting any more compared with the usual methods. However, it can be useful in some situations as we will see in the next section.

2.5 Use and Interest in Cryptography

In this section, we are dealing with elliptic curve cryptography. Elliptic curve cryptosystems were simultaneously introduced by Koblitz [14] and Miller [23]. They are becoming more and more popular because the key length can be chosen smaller than with RSA cryptosystems for the same level of security. This small key size is especially attractive for small cryptographic devices like smart cards. In all schemes (such as encryption/decryption or signature generation/verification) the dominant operation is the scalar multiplication of some

point on the elliptic curve. Hence, the efficiency of this scalar multiplication is central in elliptic curve cryptography, and more generally in cryptography based on the discrete logarithm problem. In the case where the curve is in the Montgomery form, we saw in the previous sections that the Montgomery scalar multiplication method allows to compute the multiple of any given point on the curve faster than with the usual scalar multiplication algorithms. Unfortunately, we also saw that some elliptic curves cannot be transformed into the Montgomery form. This is for example the case for most of the standards. The reason is really simple: any curve which can be transformed in the Montgomery form has a 2-torsion point so that its cardinality is divisible by 2 and this is not ideal for cryptographic use since we prefer to use curves with prime order.
In the general case, the Montgomery method can also be applied but is much more time-consuming. Indeed, we need to perform both an addition and a doubling for each bit of the exponent. This is not the case for example in the classical double and add algorithm where we only have to perform an addition every two bits on average (and even fewer with the sliding window method). Nevertheless, this particularity allows to resist to side-channel attacks on smart cards which is not the case with other algorithms.
This kind of attacks uses observations like timings [16], power consumption [17] or electromagnetic radiation [28]. They are based on the fact that addition and doubling are two different operations. In this situation, it is easy to decide, for each bit of the exponent, if the algorithm (double and add for example) is performing either a doubling (if the bit is 0) or a doubling and an addition (if the bit is 1). Hence, it is easy to recover the whole exponent (which was the secret key). Of course, various countermeasures have been proposed to secure the elliptic curve scalar multiplication against side-channel attacks [5]. For example, if one wants to protect a double and add algorithm, one can perform extra, useless, additions when the bit of the exponent is 0. In this way, for each bit of the exponent we perform both an addition and a doubling so that bits of the exponent are indistinguishable, but this is of course time consuming.
With the Montgomery scalar multiplication method, we always have to perform both an addition and a doubling for each bit of the exponent, so that this method is resistant against side-channel attacks. Therefore it is always interesting even with 19 multiplications at each step for general curves.
Of course elliptic curves in the Montgomery form are very attractive for people interested in elliptic curve cryptosystems on smart cards since, on the one hand, the scalar multiplication method is the most efficient one known to date and, on the other hand, it is resistant to side-channel attacks. That is one of the reasons why we want to generalize this method to hyperelliptic curves of genus 2.
Finally, for some cryptosystems, the x-coordinate of nP is sufficient but others, like the elliptic curve signature scheme ECDSA, require the y-coordinates. To recover it, we use the following result from [27] in the case of a curve in the Montgomery form.

Proposition 3. *Suppose that $R = P + Q$ with $P = (x_1, y_1)$, $Q = (x', y')$ and $R = (x^+, y^+)$. Then, if $y_1 \neq 0$, one has*

$$y^+ = \frac{(x'x_1 + 1)(x' + x_1 + 2A) - 2A - (x' - x_1)^2 x^+}{2By_1} \ .$$

For general curves, it is also possible to recover the y-coordinate ([1]).

In order to generalize this method to genus 2 curves, let us first recall some lowbrow background on these curves.

3 Background on Genus 2 Curves

First, let us note that every genus 2 curve is hyperelliptic, so that, in the following we will not state that the curves we are interested in are hyperelliptic. Moreover, we will concentrate on imaginary hyperelliptic curves. Since the characteristic of the field $\mathbf{K}$ has been chosen different from 2 and 5, the hyperelliptic curves we are interested in are given by an equation of the form

$$\mathcal{C} : y^2 = f(x) = x^5 + f_3 x^3 + f_2 x^2 + f_1 x + f_0 \text{ with } f_0, f_1, f_2, f_3 \in \mathbf{K} \ . \quad (1)$$

Contrary to elliptic curves, the set of points on genus 2 curves does not form a group. Thus, one can define the jacobian of $\mathcal{C}$, denoted $\mathcal{J}(\mathcal{C})$ which is the quotient of the group of divisors of degree 0 by the principal divisors. In the case of elliptic curves, this jacobian is isomorphic to the curve itself. More details on the definition of the jacobian can be found in [15]. Our purpose in this paper is to give an algorithm for scalar multiplication in this jacobian. There are mainly two means to represent elements in the jacobian. The first one is a consequence of the Riemann-Roch theorem and says that a divisor class can be represented by a couple of points ($P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$) on the curve which are conjugated over some quadratic extension of the base field $\mathbf{K}$. The second one makes explicit the correspondence of ideal classes and divisor classes and was introduced by Mumford [25]:

Theorem 1. *Let the function field be given via the irreducible polynomial $y^2 = f(x)$ where $f \in \mathbf{K}[x]$ and deg(f)=5. Each non trivial ideal class over $\mathbf{K}$ can be represented via a unique ideal generated by $u(x)$ and $y - v(x)$, $u, v \in \mathbf{K}[x]$, where u is monic, deg(v) <deg(u) ≤ 2 and $u|(v^2 - f)$.*

The correspondence between these representations is that $u(x) = (x - x_1)(x - x_2)$ and $v(x_i) = y_i$ with appropriate multiplicities. This Mumford representation was used by Cantor to develop his algorithm to compute the group law on jacobians of curves [2]. Several researchers such as Harley [12], or more recently Lange [19], [20], [21] made explicit the steps of Cantor's Algorithm and list the operations one really needs to perform. They obtained explicit formulas for the group law on the jacobian.

The basic objects are now defined and we can give an analog for genus 2 curves of the Montgomery form for elliptic curves.

4 A Montgomery-like Form for Genus 2 Curves

4.1 Definition

In the following, we will say that a curve $\mathcal{C}$ is transformable into Montgomery-like form if it is isomorphic to a curve given by an equation of the type

$$By^2 = x^5 + f_4x^4 + f_3x^3 + f_2x^2 + x \ . \tag{2}$$

It is easy to prove that a curve $\mathcal{C}$ as defined in Sect. 3 is transformable into Montgomery-like form if and only if

- the polynomial $f(x)$ has at least one root α in $\mathbf{K}$.
- the number $f'(\alpha)$ is a fourth power in $\mathbf{K}$.

Thus, as in the case of elliptic curves, not all the curves are transformable into the Montgomery-like form. Nevertheless, since the four coefficients can be chosen arbitrarily in $\mathbf{K}$, the number of such curves is about the same as for general genus 2 curves ($O(p^4)$ if $\mathbf{K} = \mathbb{F}_p$).
Please note that the first condition means that there is at least one 2-torsion element in the jacobian variety of the curve $\mathcal{C}$, so that the cardinality of the jacobian is even.

4.2 The Kummer Surface

With elliptic curves, the main idea of the Montgomery method was to avoid the computation of the y-coordinate. At first sight an analog for genus 2 curves would be to avoid the computation of the polynomial v in the Mumford representation and keep only u. But this is not satisfying since it has no mathematical sense. In fact, with elliptic curves, working only with the x-coordinate means that we identify a point and its opposite. The analog for genus 2 curves is called the Kummer surface, where a divisor and its opposite are identified. The Kummer surface is a quartic surface in $\mathbb{P}^3$. We give here the definition of the Kummer surface and its properties without demonstrations for curves in Montgomery-like form. They were obtained using the same method as that in the book of Cassels and Flynn on genus 2 curves ([3] or [8]). The Kummer surface is the image of the map

$$\begin{aligned} \kappa : J(\mathbf{K}) &\longrightarrow \mathbb{P}^3(\mathbf{K}) \\ \{(x_1, y_1), (x_2, y_2)\} &\longmapsto \left(1, x_1 + x_2, x_1x_2, \frac{F_0(x_1, x_2) - 2By_1y_2}{(x_1 - x_2)^2}\right) \ , \end{aligned}$$

with

$$F_0(x_1, x_2) = (x_1 + x_2) + 2f_2x_1x_2 + f_3(x_1 + x_2)x_1x_2 + 2f_4x_1^2x_2^2 + (x_1 + x_2)x_1^2x_2^2 \ .$$

In the following, for any divisor $\mathcal{A} \in \mathcal{J}(\mathcal{C}))$, we will denote

$$\kappa(\mathcal{A}) = (k_1(\mathcal{A}), k_2(\mathcal{A}), k_3(\mathcal{A}), k_4(\mathcal{A})) \ .$$

More precisely, the Kummer surface is the projective locus given by an equation K of degree four in the first three variables and of degree two in the last one. The exact equation can be found online [9]. In passing from the jacobian to the Kummer surface, we have lost the group structure (as was already the case with elliptic curves) but traces of it remain. For example, it is possible to double on the Kummer surface.
Nevertheless, for general divisors $\mathcal{A}$ and $\mathcal{B}$, we cannot determine the values of the $k_i(\mathcal{A}+\mathcal{B})$ from the values of the $k_i(\mathcal{A})$ and $k_i(\mathcal{B})$ since the latter do not distinguish between $\pm\mathcal{A}$ and $\pm\mathcal{B}$, and so not between $\mathcal{A}\pm\mathcal{B}$. However the values of the $k_i(\mathcal{A}+\mathcal{B})k_j(\mathcal{A}-\mathcal{B})+k_i(\mathcal{A}-\mathcal{B})k_j(\mathcal{A}+\mathcal{B})$ are well determined. We have

Theorem 2. *There are explicit polynomials φ_{ij} biquadratic in the $k_i(\mathcal{A}), k_i(\mathcal{B})$ such that projectively*

$$k_i(\mathcal{A}+\mathcal{B})k_j(\mathcal{A}-\mathcal{B})+k_i(\mathcal{A}-\mathcal{B})k_j(\mathcal{A}+\mathcal{B})=\varphi_{ij}(\mathcal{A},\mathcal{B}) \ . \tag{3}$$

Using these biquadratic forms, we can easily compute the $k_i(\mathcal{A}+\mathcal{B})$ if the $k_i(\mathcal{A}-\mathcal{B})$ are known. We can also compute the $k_i(2\mathcal{A})$ by puting $\mathcal{A}=\mathcal{B}$.

4.3 Formulas for Adding

Proposition 4. *Let $\mathbf{K}$ be a field of characteristic $k\neq 2,3$ and let $\mathcal{C}$ be a curve of genus 2 defined over $\mathbf{K}$ in the Montgomery form as defined in Sect. 4. Let $\mathcal{K}$ denote the Kummer surface of $\mathcal{C}$. Let $\mathcal{A}$ and $\mathcal{B}$ be two divisors on the jacobian of $\mathcal{C}$, $\kappa(\mathcal{A})=(k_1(\mathcal{A}),k_2(\mathcal{A}),k_3(\mathcal{A}),k_4(\mathcal{A}))$ and $\kappa(\mathcal{B})=(k_1(\mathcal{B}),k_2(\mathcal{B}),k_3(\mathcal{B}),k_4(\mathcal{B}))$ their images in the Kummer surface. Assume that the difference $\mathcal{A}-\mathcal{B}$ is known and that the last coordinate of its image in the Kummer surface is one (remember we are in $\mathbb{P}^3(\mathbf{K})$). Then we obtain the Kummer coordinates for $\mathcal{A}+\mathcal{B}$ by the following formulas :*

$$\begin{aligned}
k_1(\mathcal{A}+\mathcal{B}) &= \varphi_{11}(\mathcal{A},\mathcal{B}) \ , \\
k_2(\mathcal{A}+\mathcal{B}) &= 2\left(\varphi_{12}(\mathcal{A},\mathcal{B})-k_1(\mathcal{A}+\mathcal{B})k_2(\mathcal{A}-\mathcal{B})\right) \ , \\
k_3(\mathcal{A}+\mathcal{B}) &= k_1(\mathcal{A}-\mathcal{B})\varphi_{33}(\mathcal{A},\mathcal{B}) \ , \\
k_4(\mathcal{A}+\mathcal{B}) &= 2(\varphi_{14}(\mathcal{A},\mathcal{B})-k_1(\mathcal{A}+\mathcal{B})k_4(\mathcal{A}-\mathcal{B})) \ ,
\end{aligned}$$

where the φ_{ij} are the biquadratic forms described in Sect. 4.2.

The expressions of the $\varphi_{ij}(\mathcal{A}+\mathcal{B})$ are available by anonymous ftp [9] but require a large number of operations in the base field to be computed. The main difficulty is to find expressions which require the least possible multiplications in $\mathbf{K}$. We now give more precisely these expressions for the φ_{ij} we are interested in. For clarity we will denote $\kappa(\mathcal{A})=(k_1,k_2,k_3,k_4)$ and $\kappa(\mathcal{B})=(l_1,l_2,l_3,l_4)$.

$$\begin{aligned}
\varphi_{11}(\mathcal{A},\mathcal{B}) &= ((k_4l_1-k_1l_4)+(k_2l_3-k_3l_2))^2 \ , \\
\varphi_{12}(\mathcal{A},\mathcal{B}) &= ((k_2l_3+k_3l_2)+(k_1l_4+k_4l_1))(f_3(k_1l_3+k_3l_1)+(k_2l_4+k_4l_2))+ \\
&\quad 2(k_1l_3+k_3l_1)(f_2(k_1l_3+k_3l_1)+(k_1l_2+k_2l_1)-(k_3l_4+k_4l_3))+
\end{aligned}$$

$$
\begin{aligned}
&2f_4(k_1l_4+k_4l_1)(k_2l_3+k_3l_2)\ ,\\
\varphi_{33}(\mathcal{A},\mathcal{B}) &= ((k_3l_4-k_4l_3)+(k_1l_2-k_2l_1))^2\ ,\\
\varphi_{14}(\mathcal{A},\mathcal{B}) &= (k_1l_1-k_3l_3)(f_3((k_1l_4+k_4l_1)-(k_2l_3+k_3l_2))+\\
&\qquad 2((k_1l_2+k_2l_1)-(k_3l_4+k_4l_3))+\\
&\qquad f_2(k_4l_4+k_2l_2)+2f_4(k_1l_1-k_3l_3))+\\
&\quad (k_2l_2-k_4l_4)((k_2l_3+k_3l_2)-(k_1l_4+k_4l_1)-f_2(k_1l_1+k_3l_3))\ .
\end{aligned}
$$

4.4 Formulas for Doubling

Proposition 5. *Let* $\mathbf{K}$ *be a field of characteristic* $k \neq 2,3$ *and let* $\mathcal{C}$ *be a curve of genus 2 defined over* $\mathbf{K}$ *in the Montgomery form as defined in Sect. 4. Let* $\mathcal{K}$ *denote the Kummer surface of* $\mathcal{C}$*. Let also* $\mathcal{A}$ *be a divisor on the jacobian of* $\mathcal{C}$ *and* $\kappa(\mathcal{A}) = (k_1, k_2, k_3, k_4)$ *its image in the Kummer surface. Then we obtain the Kummer coordinates for* $2\mathcal{A}$ *(*$\kappa(2\mathcal{A}) = \delta_1, \delta_2, \delta_3, \delta_4$*) by the following formulas :*

$$
\begin{aligned}
\delta_1 &= 2\varphi_{14}(\mathcal{A},\mathcal{A})\ ,\\
\delta_2 &= 2\varphi_{24}(\mathcal{A},\mathcal{A}) + 2f_3K(\mathcal{A})\ ,\\
\delta_3 &= 2\varphi_{34}(\mathcal{A},\mathcal{A})\ ,\\
\delta_4 &= \varphi_{44}(\mathcal{A},\mathcal{A}) + 2K(\mathcal{A})\ ,
\end{aligned}
$$

where the φ_{ij} *are the biquadratic forms described in Sect. 4.2 and* K *is the equation of the Kummer surface also described in Sect. 4.2 and such that* $K(\mathcal{A}) = 0$.

This is just a consequence of Theorem 2. Let us note that in δ_2 and δ_4 we added a multiple of the equation of the Kummer surface in order to simplify the expressions as much as possible. We give now more precisely these expressions for the δ_i.

$$
\begin{aligned}
\delta_1 &= 8(k_1^2-k_3^2)(f_4(k_1^2-k_3^2)+2(k_1k_2-k_3k_4))+\\
&\quad 8(k_1k_4-k_2k_3)(k_4^2-k_2^2+f_2(k_1k_4-k_2k_3)+f_3(k_1^2-k_3^2))\ ,\\
\delta_2 &= 8(k_1^2+k_3^2-f_3k_1k_3-3k_2k_4)(k_2^2+k_4^2-f_3(k_1^2+k_3^2)+4k_1k_3)+\\
&\quad 16(k_2k_4+f_3k_1kl_3)(f_4(k_1k_2+k_3k_4)+2(k_2^2+k_4^2)+f_2(k_1k_4+k_2k_3))+\\
&\quad 32k_1k_3(4k_2k_4+f_2(k_1k_2+k_3k_4)+(f_2^2+f_4^2)k_1k_3+8f_4(k_1k_4+k_2k_3))\ ,\\
\delta_3 &= 8(k_1^2-k_3^2)(f_2(k_1^2-k_3^2)+2(k_1k_4-k_2k_3)+f_3(k_1k_2-k_3k_4))+\\
&\quad 8(k_3k_4-k_1k_2)(k_4^2-k_2^2+f_4(k_3k_4-k_1k_2))\ ,\\
\delta_4 &= (k_2^2+k_4^2)((k_2^2+k_4^2)-2f_3(k_1^2+k_3^2)-8k_1k_3)+\\
&\quad (k_1^2+k_3^2)(f_3k_1k_3+f_4(k_1k_4+k_2k_3)+2k_2k_4+f_2(k_1k_2+k_3k_4)+\\
&\qquad (f_3^2-4f_2f_4)(k_1^2+k_3^2))-8f_2f_4(k_1k_3)^2\ .
\end{aligned}
$$

5 The Montgomery Scalar Multiplication on Genus 2 Curves in Montgomery-like Form

5.1 Algorithm

We give here an analog for genus 2 curves of the Montgomery method for scalar multiplication on elliptic curves described in Sect. 2.1. In the case of elliptic curves, Montgomery's method [24] avoids the computation of the y-coordinate. We saw that an equivalent method in genus 2 was to work on the Kummer surface. Of course we have the same restriction in the case of genus 2 curves, namely that it is not possible to add two divisors except if their difference is known. If $\mathcal{D}$ is some divisor, recalling that our goal is the computation of $n\mathcal{D}$ for some integer n, the principle is, as it was already the case for elliptic curves, to use pairs of consecutive powers of $\mathcal{D}$, so that the difference between the two components of the pair is always known and equal to $\mathcal{D}$. The algorithm for scalar multiplication is as follows:

Algorithm 2. *Montgomery scalar multiplication algorithm for genus 2 curves*
Input : $\mathcal{D} \in \mathcal{J}(\mathbf{K})$ *and* $n \in \mathbb{Z}$.
Output : $\kappa(n\mathcal{D})$*, the image in the Kummer surface of* $n\mathcal{D}$.

Step 1. Initialize $(\mathcal{A}, \mathcal{B}) = ((0,0,0,1), \kappa(\mathcal{D}))$ *where* $(0,0,0,1)$ *is the image in the Kummer surface of the neutral element on* $\mathcal{J}(\mathcal{C})$.

Step 2. If the bit of n *is 0,* $(\mathcal{A}, \mathcal{B}) = (2\mathcal{A}, \mathcal{A} + \mathcal{B})$.

Step 3. If the bit of n *is 1,* $(\mathcal{A}, \mathcal{B}) = (\mathcal{A} + \mathcal{B}, 2\mathcal{B})$.

Step 4. After doing that for each bit of n*, return* $\mathcal{A}$.

Note that, at each step, we always have $\mathcal{B} - \mathcal{A} = \kappa(\mathcal{D})$ so that the addition of $\mathcal{A}$ and $\mathcal{B}$ is possible.

5.2 Number of Operations

At each step of the algorithm, we perform both an addition and a doubling, hence we just have to count the number of operations required for each of them. In the following, M will denote a multiplication in $\mathbf{K}$ and S a squaring.

Table 1. Addition of $\mathcal{A}$ and $\mathcal{B}$ in $\mathcal{K}(\mathcal{C})$ if $\mathcal{A} - \mathcal{B}$ is known

expressions	operations
precomputations $\{k_i l_j\}_{i,j=1..4}$	$16M$
$\varphi_{11}(\mathcal{A}, \mathcal{B})$	S
$\varphi_{12}(\mathcal{A}, \mathcal{B})$	$6M$
$\varphi_{33}(\mathcal{A}, \mathcal{B})$	S
$\varphi_{14}(\mathcal{A}, \mathcal{B})$	$6M$
$\kappa(\mathcal{A} + \mathcal{B})$	$3M$
total	$31M + 2S$

Remark 1. The 31 multiplications include 7 multiplications by coefficients of the curve.

Table 2. Doubling of $\mathcal{A}$ in $\mathcal{K}(\mathcal{C})$

expressions	operations
precomputations $\{k_i k_j\}_{i,j=1..4}$	$6M + 4S$
$\delta_1(\mathcal{A}, \mathcal{B})$	$5M$
$\delta_2(\mathcal{A}, \mathcal{B})$	$11M$
$\delta_3(\mathcal{A}, \mathcal{B})$	$5M$
$\delta_4(\mathcal{A}, \mathcal{B})$	$5M$
total	$31M + 5S$

Remark 2. The 31 multiplications include 16 multiplications by coefficients of the curve. Moreover the multiplications $f_3 k_1 k_3$, $f_3(k_1^2 + k_3^2)$, $f_4(k_1 k_4 + k_2 k_3)$ and $f_2(k_1 k_2 + k_3 k_4)$ are not counted in δ_4 since they were already computed in δ_2. Finally, we of course assumed that $f_2 f_4$, $f_3^2 - 4 f_2 f_4$ and $f_2^2 + f_4^2$ were precomputed.

Hence, on a curve in the Montgomery form as in (2), the scalar multiplication using the Montgomery method requires $69|n|_2$ base field multiplications (assuming that a squaring is a multiplication), where $|n|_2$ is the number of bits of n.

5.3 Comparison with Usual Algorithms for Scalar Multiplication

To date, the best algorithms for scalar multiplication on genus 2 curves defined over a field of odd characteristic are obtained by using mixed weighted projective coordinates [21]. In this case, Lange needs 41 multiplications both for a mixed addition and for a doubling. Hence our formulas requires fewer base field operations. But, in the Montgomery algorithm, we must perform both an addition and a doubling for each bit of the exponent whereas one can use efficient algorithms (like the sliding window method) with Lange's formulas. Nevertheless, this algorithm is still interesting for many reasons.

- As was the case for elliptic curves and as explained in Sect. 2.5, the Montgomery algorithm is resistant to side-channel attacks, contrary to other algorithms for scalar multiplications. For this reason it will be of interest to people who need to implement hyperelliptic curves protocols on smart cards or systems sensitive to side-channel attacks. For example, if one wants to make safe algorithms using mixed weighted projective coordinates, one needs to perform an extra addition when the bit of the exponent is one. In this case, for each bit of the exponent, 82 base field operations are required and with only 69, our algorithm allows a gain of 16 percent, which is significant.
- This algorithm is very easy to implement, there are no precomputations (as in the sliding window method) and an element on the Kummer surface requires only 4 base field elements whereas weighted projective coordinates require 8 of them so that it is also interesting in terms of memory usage.

This last remark will be an advantage for constrained environments like smart cards.
– It is very dependent of the coefficients of the curve. Indeed there are 23 multiplications by these coefficients but only 2 in Lange's formulas. Hence a good choice of the coefficients of the curve certainly allows better timings. This is the subject of the following section.

5.4 Some Special Cases

In order to decrease the number of base field operations for our algorithm, certain choices of coefficients of the curve are better to use. For example there are 6 multiplications by f_3 in the formulas given in Sects. 4.3 and 4.4 so that, if one chooses $f_3 = 0$ or 1, the total amount of multiplications necessary for each bit of the exponent is 63 instead of 69. In the following table, we summarize the gain obtained in each operation. Let us note that there is no gain for the calculation of φ_{11}, φ_{33} and precomputations.

Table 3. Gain obtained if ...

	$f_2 = 0$	f_2 small	$f_3 = 0$ or small	$f_4 = 0$	f_4 small
φ_{12}	1	1	1	2	1
φ_{14}	2	2	1	1	1
δ_1	1	1	1	1	1
δ_2	2	2	2	2	2
δ_3	1	1	1	1	1
δ_4	2	1	0	2	1
total	9	8	6	9	7

Remark 3. If two of these conditions on the coefficients are satisfied the gain obtained is just the sum of the gains.

Of course this kind of restriction implies that fewer curves are taken into account. For example, if $\mathbf{K} = \mathbb{F}_p$ and $f_3 = 0$, one can only choose three coefficients in $\mathbb{F}_p$ (namely f_2, f_4 and B) so that the number of such curves is $O(p^3)$. Thus we lose in generality. However, in cryptography, one only needs to find a curve such that the order of its jacobian is divisible by a huge prime number. For this, one needs enough choices of curves in order to be able to find a curve with this property and $O(p^3)$ choices are of course widely sufficient.
Let us now examine more precisely a particular case and compare our algorithm to usual ones. Let $\mathcal{C}$ be a genus 2 curve defined over $\mathbf{K}$ by an equation of the form

$$By^2 = x^5 + f_3x^3 + \varepsilon x^2 + x \text{ with } \varepsilon = 0 \text{ or } \pm 1 \text{ and } B \text{ and } f_3 \in \mathbf{K} \ . \quad (4)$$

There are $O(p^2)$ curves in this form (which is sufficient to find one of these with nice properties for use in cryptography). Here, our algorithm of scalar multiplication requires 52 multiplications for each bit of the exponent whereas with mixed weighted projective coordinates,

– a sliding window method with window size equal to 4 requires in average 48 multiplications,
– a classical double and add requires 61 multiplications on average,
– a side-channel attack resistant double and add requires 81 multiplications.

Thus, our algorithm is 15 percent faster than a double and add, not so far from the sliding window method (around 7 percent) and much more efficient if one wants the operation to be resistant to side-channel attacks. Indeed, in this case, we obtain a gain of 36 percent. Of course one can even be faster than the sliding window method by choosing a small coefficient f_3 but the number of such curves becomes small.

Remark 4. Another means to accelerate this algorithm would be to choose f_2, f_3 and f_4 one word long. For example, on a 32 bits processor, if we are working on some finite field of cryptographic size for genus 2 curves, a multiplication of a coefficient of the curve and an element of the base field is about three times faster than the usual multiplication in the base field. Hence, as there are 23 multiplications by coefficients of the curve, our algorithm will require the equivalent of 53 multiplications, which is not so bad.

5.5 Examples

In this section, the base field is the prime field $\mathbb{F}_{2^{80}+13}$ (so that cryptosystems based on genus 2 curves defined over this field have the same security level than those based on elliptic curves defined over some 160 bits prime field). Let $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ be the genus 2 curves respectively defined by the equations

$$\begin{aligned}
442946377804223815965 77\, y^2 &= x^5 + 271937476575616687835 34\, x^4 \\
&\quad + 297559220985752192390 37\, x^3 \\
&\quad + 760478620401411267378 26\, x^2 + x \ , \\
103779310474567225222 92\, y^2 &= x^5 + 773041988679881578656 77\, x^3 + x^2 + x \ , \\
694188370684424938642 20\, y^2 &= x^5 + x^3 + x \ .
\end{aligned}$$

We compared our algorithm on these curves with a sliding window of size 4, a classical double and add and a double and always add (used to resist against side-channel attacks). For these three algorithms, we of course always used the weighted projective coordinates as in [21] which are the more efficient ones. In the following table, we provide the timings obtained using GMP 4.1.2 on a Pentium IV 3.06 GHz. We carried out 1000 scalar multiplications in each case with various divisors and 160 bits exponents.

Table 4. Timings

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$
Sliding window	13.4 ms	13.3 ms	12.9 ms
Double and add	16 ms	16 ms	15.5 ms
Double and always add	21.5 ms	21.5 ms	21 ms
Montgomery method	18.3 ms	13.6 ms	11.9 ms

6 Conclusion and Prospects

Thanks to the theory of the Kummer surface of a hyperelliptic curve of genus 2, we have generalized to genus 2 curves the method of Montgomery for scalar multiplication on elliptic curves. As Montgomery does for elliptic curves, we restrict to curves transformable into Montgomery-like form. However, there are no theoretical obstructions to generalize this method to all genus 2 curves. Indeed Propositions 4.3 and 4.4 remain valid but the total amount of multiplications to compute the biquadratic forms is really huge so that this method is not competitive with the classical ones. This is not so surprising since it was already the case for elliptic curves.

In fact, for people interested in cryptography, this restriction is not very important since the number of choices of curves remains largely the same. The only significant restriction is that the order of the jacobian of such curves is even and then cannot be prime. But working with a jacobian whose order is twice a prime is not less efficient than working with a prime order.

For elliptic curves, the standards are not transformable into the Montgomery form because of this restriction and it's really a shame because the Montgomery method for scalar multiplication is the most interesting one (the fastest, easy to implement, resistant to side-channel attacks). Up to now, there are no standards for genus 2 curves. If such standards exist one day, it would be useful to take the method that we developed into account.

Moreover, we have seen that, with some restrictions, we obtain very interesting timings for the scalar multiplication on the jacobian of genus 2 curves. It would be nice to verify (even if there is no reason for this) that these restrictions are not awkward for finding jacobians suitable for cryptography (i.e. with a large prime dividing the order). Unfortunately, algorithms for finding the order of the jacobian over $\mathbb{F}_p$ are still under development ([11], [22]).

Finally, it would be very interesting to study the case of the characteristic 2, since it is in that case that this method is the most efficient for elliptic curves. For this, all the necessary mathematical objects, such as the Kummer surface, remain to be defined.

References

1. Brier, E., Joye, M.: Weierstrass Elliptic Curves and Side-Channel Attacks, Public Key Cryptography, Lecture Notes in Computer Science, **2274** (2002)
2. Cantor, D. G.: Computing on the Jacobian of a hyperelliptic curve. Math. Comp., **48** (1987) 95–101
3. Cassel, J. W. S., Flynn, E. V.: Prolegomena to a middlebrow arithmetic of curves of genus 2, London Mathematical Society Lecture Note Series, **230** (1996)
4. Cohen, H., Miyaji, A., Ono, T.: Efficient elliptic curve exponentiation using mixed coordinates, Asiacrypt'98, Lecture Notes in Computer Science, **1514** (1998) 51–65
5. Coron, J. S.: Resistance against differential power analysis for elliptic curve cryptosystems, CHES'99, Lecture Notes in Computer Science, **1717** (1999) 292–302

6. Doche, C., Duquesne, S.: Manual for the elliptic curve library, Arehcc report (2003)
7. Duquesne, S.: Improvement of the Montgomery method for general elliptic curves defined over $\mathbb{F}_p$, preprint (2003)
8. Flynn, E. V.: The group law on the Jacobian of a curve of genus 2, J. reine angew. Math., **439** (1993), 45–69
9. Flynn, E. V.: ftp site, ftp://ftp.liv.ac.uk/pub/genus2/kummer
10. Fischer, W., Giraud, C., Knudsen, E. W., Seifert, J. P.: Parallel scalar multiplication on general elliptic curves over $\mathbb{F}_p$ hedged against Non-Differential Side-Channel Attacks, preprint
11. Gaudry, P., Schost, E.: Construction of secure random curves of genus 2 over prime fields, Eurocrypt'04, Lecture Notes in Computer Science (2004)
12. Harley, R.: Fast arithmetic on genus 2 curves, available at http://cristal.inria.fr/∼harley/hyper (2000)
13. Izu, T., Takagi, T.: A fast Elliptic Curve Multiplication Resistant against Side Channel Attacks, preprint
14. Koblitz, N.: Elliptic curve cryptosystems, Math. Comp., **48** (1987) 203–209
15. Koblitz, N.: Algebraic aspects of cryptography, Algorithms and Computation in Mathematics, **3** (1998)
16. Kocher, P. C.: Timing attacks on implementations of DH, RSA, DSS and other systems, CRYPTO'96, Lecture Notes in Computer Science, **1109** (1996), 104–113
17. Kocher, P. C., Jaffe, J., Jun, B.: Differential power analysis, CRYPTO'99, Lecture Notes in Computer Science, **1666** (1999) 388-397
18. Koyama, K., Tsuruoka, Y.: Speeding up elliptic cryptosystems by using a signed binary window method, Crypto'92, Lecture Notes in Computer Science, **740** (1993) 345–357
19. Lange, T.: Efficient Arithmetic on Genus 2 Hyperelliptic Curves over Finite Fields via Explicit Formulae, Cryptology ePrint Archive, **121** (2002)
20. Lange, T.: Inversion-Free Arithmetic on Genus 2 Hyperelliptic Curves, Cryptology ePrint Archive, **147** (2002)
21. Lange, T.: Weighted Coordinates on Genus 2 Hyperelliptic Curves, Cryptology ePrint Archive, **153** (2002)
22. Matsuo, K. Chao, J. Tsujii, S.: An improved baby step giant step algorithm for point counting of hyperelliptic curves over finite fields, ANTS-V, Lecture Notes in Computer Science, **2369** (2002) 461–474
23. Miller, V. S.: Use of elliptic curves in cryptography, Crypto'85, Lecture Notes in Computer Science, **218** (1986) 417–426
24. Montgomery, P. L.: Speeding the Pollard and elliptic curve methods of factorization, Math. Comp., **48** (1987) 243–164
25. Mumford, D.: Tata lectures on Theta II, Birkhuser (1984)
26. Okeya, K., Kurumatani, H., Sakurai, K.: Elliptic curves with the Montgomery-form and their cryptographic applications, Public Key Cryptography, Lecture Notes in Computer Science, **1751** (2000) 238–257
27. Okeya, O., Sakurai, K.: Efficient Elliptic Curve Cryptosystems from a Scalar Multiplication Algorithm with Recovery of the y-Coordinate on a Montgomery-Form Elliptic Curve, Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science, **2162** (2001) 126–141
28. Quisquater, J. J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards, e-smart 2001, Lecture Notes in Computer Science, **2140**, (2001), 200–210