Montgomery Ladder for all Genus 2 Curves in Characteristic 2

Sylvain Duquesne

Université Montpellier II,
Laboratoires I3M, UMR CNRS 5149 and LIRMM, UMR CNRS 5506
Place Eugène Bataillon CC 051, 34005 Montpellier Cedex, France
duquesne@math.univ-montp2.fr

Abstract. Using the Kummer surface, we generalize Montgomery ladder for scalar multiplication to the Jacobian of genus 2 curves in characteristic 2. Previously this method was known for elliptic curves and for genus 2 curves in odd characteristic. We obtain an algorithm that is competitive compared to usual methods of scalar multiplication and that has additional properties such as resistance to side-channel attacks. Moreover it provides a significant speed-up of scalar multiplication in many cases. This new algorithm has very important applications in cryptography using hyperelliptic curves and more particularly for people interested in cryptography on embedded systems (such as smart cards).

Key words: Hyperelliptic curves, Characteristic 2, Kummer surface, Cryptography, Scalar multiplication.

1 Introduction

Elliptic curve cryptosystems were simultaneously introduced by Koblitz [15] 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. Hyperelliptic curves allow to generalize elliptic curves cryptosystems on smaller base field where basic operations are cheaper. In all schemes, the dominant operation is a scalar multiplication of some point on an elliptic curve (or some element on the Jacobian of a hyperelliptic curve). Hence, the efficiency of this scalar multiplication is central in elliptic and hyperelliptic curve cryptography. In this paper we are dealing with scalar multiplication on Jacobian of genus 2 curves defined over a field of characteristic 2.

For certain elliptic curves, Montgomery [24] developed a method, called Montgomery ladder, allowing faster scalar multiplication than usual methods. His method has the extra advantage that it is resistant to side-channel attacks. This is very interesting for people who want to use elliptic curves on embedded devices like smart cards. This method was generalized to any elliptic curves in

odd characteristic in [1], to elliptic curves in characteristic 2 [22] and more recently to genus 2 curves in odd characteristic [6, 11]. Finally, Gaudry announced last year in [12] he was able to deal with certain genus 2 curves in characteristic 2 and he is currently writing a paper on the subject with Lubicz.

The aim of this paper is the generalization to all genus 2 curves in characteristic 2. In the following, $\mathbf{K}$ will denote a field of characteristic 2, M a multiplication in $\mathbf{K}$ and S a squaring. For cryptographic applications, the base field we have in mind is $\mathbb{F}_{2^d}$ where d is a prime number (because of the Weil descent [10]). This paper is organized as follows: in Sections 2 and 3 we recall Montgomery ladder for elliptic curves and basic statements on genus 2 curves. In Section 4 we introduce the Kummer surface in characteristic 2 which allows to develop a Montgomery ladder. Formulas for addition and doubling and comparisons with other formulas are given in Section 5.

2 Montgomery Ladder on Elliptic Curves in Characteristic 2

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

$$y^2 + xy = x^3 + a_2x^2 + a_6 \ ,$$

with $a_6 \neq 0$. Every non-supersingular elliptic curve defined over $\mathbf{K}$ is isomorphic to a curve given by such an equation. The problem we are interested in for cryptographic purposes 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 methods to do this (double and add, sliding window, w-NAF, ...). To improve these algorithms, it is very convenient to use projective coordinates in order to avoid expensive divisions in $\mathbf{K}$. Hence, we use a triple (X, Y, Z) in $\mathbb{P}^2(\mathbf{K})$ such that $x = X/Z$ and $y = Y/Z$ to represent the point (x, y).

In [24], Montgomery proposed to avoid computation of the y-coordinate, so that we can hope that basic operations (doubling and addition) are easier to compute. Since, for any x-coordinate, there are two corresponding points on the curve which are opposite, this restriction is equivalent to identifying a point on the curve and its opposite. If we want to add two points $\pm P$ and $\pm Q$, we cannot decide if the result obtained is either $\pm(P+Q)$ as required or $\pm(P-Q)$. So the group law is lost. Nevertheless, some operations remain possible like doubling since it is easy to decide if the result is $\pm 2P$ or the point at infinity. Unfortunately, doubling is not sufficient for a complete scalar multiplication. In fact additions are possible if the difference $P - Q$ is known. Then, the principle to compute nP is to use pairs of consecutive multiples of P. The algorithm for scalar multiplication, usually called Montgomery ladder, 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,* $Q = (2Q_1, Q_1 + Q_2)$.

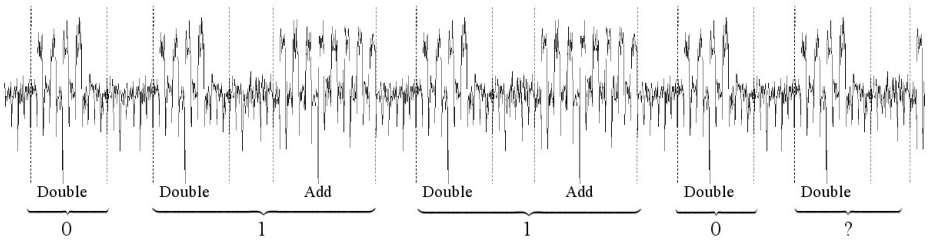
Step 3. If the bit of n *is 1,* $Q = (Q_1 + Q_2, 2Q_2)$.

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

At each step, $Q = (kP, (k+1)P)$ for some k and we compute either $(2kP, (2k+1)P)$ or $((2k+1)P, (2k+2)P)$, so we always have $Q_2 - Q_1 = P$ and additions can be performed.
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 pay to avoid the y-coordinate but we hope that the gain obtained thanks to this restriction will be sufficient to compensate for the larger number of operations.

In [22], Lopez and Dahab generalized Montgomery's idea to binary curves and gave formulas for addition (assuming the difference is known) and doubling requiring respectively 4 multiplications and 1 squaring in $\mathbf{K}$ (4M and 1S) and 2 M and 3 S. Thus, the cost of Montgomery ladder is about $6|n|_2$ M and $4|n|_2$ S where $|n|_2$ denotes the number of bits of n. In the best case, usual scalar multiplication requires 5 M and 4 S for doubling and 9 M and 5 S for addition (Lopez-Dahab coordinates [21]). So, even if the algorithm used involved very few additions, Montgomery ladder is more efficient. For instance, the sliding window method with window size 4 requires around $7|n|_2$ M and $5|n|_2$ S. Moreover, Montgomery ladder has the extra advantage to be resistant against simple side-channel attacks.

These attacks use observations like timings [17], power consumption [18] or electromagnetic radiation [27]. They are based on the fact that addition and doubling are two different operations on elliptic curves. It is then 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 is often 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 we want to protect a double and add algorithm, we 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 bits of the exponent are indistinguishable. This is of course time consuming and sensible to fault attacks.

With Montgomery ladder, we always have to perform both an addition and a doubling for each bit of the exponent, so this method is naturally resistant to side-channel attacks. Therefore it is particularly interesting and attractive for people interested in elliptic curve cryptosystems on embedded systems. That is one of the reasons why we want to generalize this method to hyperelliptic curves of genus 2.

Finally, for some cryptosystems (like Diffie-Hellman key exchange, authentication), the x-coordinate of nP is sufficient but others require the y-coordinate. Formulas to recover it are given in [22, 26]. However, it is also possible to use variants of these protocols where the x-coordinate is sufficient. Of course these remarks are also valid for hyperelliptic curves [28]. Thereafter we will not take this into consideration.

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

3 Background on Genus 2 Curves in Characteristic 2

Every genus 2 curve is hyperelliptic. So, in the following, we do not state that the curves we are interested in are hyperelliptic.
Moreover, as usual in cryptography, we will concentrate on imaginary curves defined over $\mathbf{K}$ which are given by equations of the form

$$\mathcal{C} : y^2 + h(x)y = f(x) \ . \tag{1}$$

where f and g are in $\mathbf{K}[x]$, $\deg(f) = 5$, $\deg(h) \leq 2$ and there is no singular points.

3.1 Arithmetic of Genus 2 Curves

Contrary to elliptic curves, the set of points on genus 2 curves does not form a group. But the Jacobian of $\mathcal{C}$, denoted $\mathcal{J}(\mathcal{C})$, is a group (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 [16]. There are mainly two ways to represent elements in the Jacobian:

- with 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 $\mathbf{K}$ (this is a consequence of the Riemann-Roch theorem),

- with 2 polynomials u and v in $\mathbf{K}[x]$ where u is monic, $\deg(v) < \deg(u) \leq 2$ and $u|(v^2 + hv + f)$ (Mumford representation [25]).

The correspondence between these representations is that $u(x) = (x+x_1)(x+x_2)$ and $v(x_i) = y_i$ with appropriate multiplicities. In [3], Cantor described the group law on Jacobians with Mumford representation. Several researchers such as Harley [13], or more recently Lange [20] made explicit the steps of Cantor's Algorithm and listed the operations one really needs to perform. They obtained explicit formulas for the group law on the Jacobian which are an analog of the different choices of coordinates for elliptic curves. In this this paper, our purpose is to give an analog of Montgomery ladder. Let us first recall that the equation defining the curve can be simplified.

3.2 Classification of Genus 2 Hyperelliptic Curves over $\mathbb{F}_{2^d}$

As explained in the previous section, we are interested in curves defined over $\mathbb{F}_{2^d}$ by an equation of the form

$$y^2 + (h_2x^2 + h_1x + h_0)y = x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0 \qquad (2)$$

In fact, genus 2 curves can be divided into three types depending on the leading coefficient of h. Following the notations of [4], we have:

- type I: $h_2 \neq 0$.
- type II: $h_2 = 0$, $h_1 \neq 0$.
- type III: $h_2 = h_1 = 0$, $h_0 \neq 0$.

Moreover, Choie and Yun prove in [4] that type I has asymptotically between $2q^3$ and $4q^3$ isomorphism classes ($q = 2^d$), type II about $2q^2$ and type III between $2q$ and $32q$. However, Galbraith proved in [9] that type III curves are supersingular (hence cryptographically weaker), so only curves of types I and II are interesting for cryptosystems.

In [2], equations for type I and type II are given in a minimal form, in the sense that if the coefficients range through the base field, the expected number of curves is obtained (say $4q^3$ for type I and $2q^2$ for type II). We recall these minimal forms with slight modifications (we perform a very easy change of variable to assume that $f_1 = 0$ instead of $f_2 = 0$ and we exploit that d is odd). In the following, ε is an element of $\mathbb{F}_2$.

Theorem 1. *A hyperelliptic curve of type I defined over $\mathbb{F}_{2^d}$ with d odd can be transformed into one of the following equations:*

$$\begin{aligned} &\textit{type Ia} : y^2 + (x^2 + h_1x + h_1^2)y = x^5 + \varepsilon x^4 + f_2x^2 + f_0, \\ &\textit{type Ib} : y^2 + (x^2 + h_1x)y = x^5 + \varepsilon x^4 + f_2x^2 + f_0. \end{aligned}$$

A hyperelliptic curve of type II defined over $\mathbb{F}_{2^d}$ with d odd can be transformed into the following equation :

$$y^2 + xy = x^5 + f_3x^3 + \varepsilon x^2 + f_0.$$

Remark 1. It is shown in [2] that we can easily find curves of each type suitable for cryptography. Moreover, it will be interesting in the following to choose $h_1 = 1$ for curves of type I. Even with this restriction, it remains sufficiently many isomorphism classes for curves of type I ($4q^2$) and, again, there is no obstruction to find such curves suitable for cryptography.

4 The Kummer Surface in Characteristic 2

With elliptic curves, the main idea of Montgomery method was to avoid the computation of the y-coordinate. It has already been explained in [6] (in odd characteristic) that a good way to generalize this idea is to use the Kummer surface. This object is the quotient of the Jacobian by the hyperelliptic involution. In other words, we identify an element of the Jacobian and its opposite. The Kummer surface is a quartic surface in $\mathbb{P}^3$. We give here the definition of the Kummer surface and its properties. Proofs and more details can be found in [7]. The Kummer surface is the image of the map

$$\begin{aligned} \kappa : \mathcal{J}(\mathbf{K}) &\longrightarrow \mathbb{P}^3(\mathbf{K}) \\ \{(x_1, y_1), (x_2, y_2)\} &\longmapsto [k_1, k_2, k_3, k_4] \end{aligned}$$

$$\begin{aligned} \text{with } k_1 &= 1 \\ k_2 &= x_1 + x_2 \\ k_3 &= x_1x_2 \\ k_4 &= \frac{(x_1 + x_2)\left(x_1^2x_2^2 + f_3x_1x_2 + f_1\right) + h(x_2)y_1 + h(x_1)y_2}{(x_1 + x_2)^2} \end{aligned}$$

Let us note that the image of the point at infinity is $[0, 0, 0, 1]$. 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 in [7]. In passing from the Jacobian to the Kummer surface, we lost the group structure 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 values of $k_i(\mathcal{A} + \mathcal{B})$ from values of $k_i(\mathcal{A})$ and $k_i(\mathcal{B})$ since the latter does not distinguish between $\pm\mathcal{A}$ and $\pm\mathcal{B}$, and so not between $\pm(\mathcal{A} + \mathcal{B})$ and $\pm(\mathcal{A} - \mathcal{B})$ (as was already the case with elliptic curves). However values of $k_i(\mathcal{A} + \mathcal{B})k_j(\mathcal{A} - \mathcal{B}) + \varepsilon_{ij}k_i(\mathcal{A} - \mathcal{B})k_j(\mathcal{A} + \mathcal{B})$ are well determined. We have [7]

Theorem 2. *Let $\mathcal{A}$, $\mathcal{B}$ in $\mathcal{J}(\mathbf{K})$ and $\kappa(\mathcal{A})$, $\kappa(\mathcal{B})$ their image in the Kummer surface. Then, for $i, j \in \{1, \ldots, 4\}$, there are explicit polynomials φ_{ij} biquadratic in $k_i(\mathcal{A}), k_i(\mathcal{B})$ such that projectively*

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

where $\varepsilon_{ij} = 1$ *if* $i \neq j$ *and* 0 *if* $i = j$.

These traces of the group law allow to give an analog for genus 2 curves of Montgomery ladder. Let $\mathcal{D}$ be an element of the Jacobian. Our purpose is the computation of $n\mathcal{D}$ for some integer n. As it is not possible to add two divisors except if their difference is known, the principle is (as for elliptic curves) to use pairs of consecutive multiples 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 $(\mathcal{A}, \mathcal{B}) = ([0, 0, 0, 1], \kappa(\mathcal{D}))$.

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* $\mathcal{A}$.

Let us now explain more precisely how the addition and the doubling can be derived from the biquadratic forms and give explicit formulas.

5 Formulas for Addition and Doubling

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

Proposition 1. *Let* $\mathbf{K}$ *be a field of characteristic* 2 *and let* $\mathcal{C}$ *be a curve of genus 2 defined over* $\mathbf{K}$ *by an equation of the form (2). Let* $\mathcal{A}, \mathcal{B} \in \mathcal{J}(\mathcal{C})$ *and* $\kappa(\mathcal{A}) = [k_1(\mathcal{A}), k_2(\mathcal{A}), k_3(\mathcal{A}), k_4(\mathcal{A})], \kappa(\mathcal{B}) = [k_1(\mathcal{B}), k_2(\mathcal{B}), k_3(\mathcal{B}), k_4(\mathcal{B})]$ *their images in the Kummer surface. Assume that the difference* $\mathcal{A}-\mathcal{B}$ *is known and that* $k_1(\mathcal{A}-\mathcal{B}) = 1$ *(remember we are in* $\mathbb{P}^3(\mathbf{K})$*). Then we obtain the Kummer coordinates for* $\mathcal{A} + \mathcal{B}$ *by the following formulas :*

$$\begin{aligned}
k_1(\mathcal{A} + \mathcal{B}) &= \varphi_{11}(\mathcal{A}, \mathcal{B}) \\
k_2(\mathcal{A} + \mathcal{B}) &= \varphi_{12}(\mathcal{A}, \mathcal{B}) + k_1(\mathcal{A} + \mathcal{B}) \times k_2(\mathcal{A} - \mathcal{B}) \\
k_3(\mathcal{A} + \mathcal{B}) &= \varphi_{13}(\mathcal{A}, \mathcal{B}) + k_1(\mathcal{A} + \mathcal{B}) \times k_3(\mathcal{A} - \mathcal{B}) \\
k_4(\mathcal{A} + \mathcal{B}) &= \varphi_{14}(\mathcal{A}, \mathcal{B}) + k_1(\mathcal{A} + \mathcal{B}) \times k_4(\mathcal{A} - \mathcal{B}).
\end{aligned}$$

The formulas for $2\mathcal{A}$ *are*

$$\begin{aligned}
k_1(2\mathcal{A}) &= \varphi_{14}(\mathcal{A}, \mathcal{A}) \\
k_2(2\mathcal{A}) &= \varphi_{24}(\mathcal{A}, \mathcal{A}) \\
k_3(2\mathcal{A}) &= \varphi_{34}(\mathcal{A}, \mathcal{A}) \\
k_4(2\mathcal{A}) &= \varphi_{44}(\mathcal{A}, \mathcal{A}).
\end{aligned}$$

The expressions of the φ_{ij} are given in [7] and are available on the web page of the author [8]. However, they require a large number of operations for a curve given by the general equation (2). The main difficulty is to find expressions which require the least possible multiplications in $\mathbf{K}$. In order to provide formulas as efficient as possible, we distinguish the types of curve. In each case, we give more precise expressions of the φ_{ij} we are interested in. For clarity we denote $\kappa(\mathcal{A}) = (k_1, k_2, k_3, k_4)$ and $\kappa(\mathcal{B}) = (l_1, l_2, l_3, l_4)$. We also use the sign $\times$ to denote multiplications that must be done and nothing for multiplications already done before in the formulas. We also choose to provide compact formulas but step-by-step (verified) formulas are given on the web page of the author [8].

5.1 Formulas for Curves of Type Ia

Remember that any curve of type Ia can be defined by an equation of the form

$$y^2 + (x^2 + h_1 x + h_1^2)y = x^5 + \varepsilon x^4 + f_2 x^2 + f_0.$$

Addition and doubling can be computed using the following formulas for φ_{ij} :

$$\begin{aligned}
\varphi_{11}(\mathcal{A},\mathcal{B}) &= (k_1{\times}l_4 + k_2{\times}l_3 + k_4{\times}l_1 + k_3{\times}l_2)^2\\
\varphi_{12}(\mathcal{A},\mathcal{B}) &= \big(k_1l_4 + k_4l_1 + h_1{\times}k_1{\times}l_3 + h_1{\times}k_3{\times}l_1 + h_1^2{\times}k_1{\times}l_2 + h_1^2{\times}k_2{\times}l_1\big)\times\\
&\qquad \big(k_2l_3 + k_3l_2 + h_1k_1l_3 + h_1k_3l_1 + h_1^2k_1k_2 + h_1^2k_2l_1\big)\\
\varphi_{13}(\mathcal{A},\mathcal{B}) &= h_1^2{\times}(k_4l_1 + k_1l_4){\times}(h_1k_1{\times}h_1l_1 + k_2{\times}l_2) + (h_1k_3l_1 + k_4l_1){\times}(k_3{\times}l_3 +\\
&\quad h_1{\times}(h_1k_3l_1 + k_2l_3 + h_1^2k_1l_2)) + (h_1k_1l_3 + k_1l_4){\times}(k_3l_3 + h_1{\times}(h_1k_1l_3 +\\
&\quad k_3l_2 + h_1^2k_2l_1)) + (h_1^2k_2l_1 + h_1^2k_1l_2){\times}\alpha\\
\varphi_{14}(\mathcal{A},\mathcal{B}) &= (\alpha + h_1{\times}k_1l_4)^2 + k_3l_2{\times}(h_1^2{\times}(k_4l_1 + h_1^2k_2l_1 + h_1^2k_1l_2 + k_2l_3) + \beta) +\\
&\quad (k_4l_1 + k_1l_4 + k_2l_3){\times}(h_1^2{\times}(k_1l_4 + h_1^2k_2l_1 + h_1^2k_1l_2) + \beta)\\
\text{with } \alpha &= h_1{\times}(k_2l_3 + k_3l_2 + h_1k_1l_3 + h_1k_3l_1) + k_3l_3 + h_1^2{\times}k_2l_2\\
\beta &= h_1{\times}(k_3l_3 + h_1^2k_2l_2)
\end{aligned}$$

$$\begin{aligned}
\varphi_{14}(\mathcal{A},\mathcal{A}) &= \alpha + \beta^2\\
\varphi_{24}(\mathcal{A},\mathcal{A}) &= h_1{\times}((h_1^2{\times}k_1^2 + k_2^2){\times}(d_1{\times}k_1^2 + h_1^2{\times}d_2{\times}k_1^2 + k_4^2) + (d_2k_1^2 + h_1^2{\times}k_2^2){\times}\beta +\\
&\quad \varphi_{14}(\mathcal{A},\mathcal{A}))\\
\varphi_{34}(\mathcal{A},\mathcal{A}) &= \beta{\times}(k_4^2 + h_1^2d_2k_1^2) + h_1^2{\times}(d_1k_1^2{\times}(h_1^2k_1^2 + \tfrac{1}{h_1^2}{\times}k_3^2) + \alpha)\\
\varphi_{44}(\mathcal{A},\mathcal{A}) &= h_1{\times}(h_1^2k_2^2{\times}(d_1k_1^2 + h_1^2d_2k_1^2 + c_2{\times}k_2^2) + k_3^2{\times}(k_4^2 + d_1k_1^2 + c_3{\times}k_3^2)) +\\
&\quad (c_1{\times}k_1^2 + k_4^2)^2\\
\text{with } \alpha &= h_1^2{\times}(k_1^2{\times}k_4^2 + k_2^2{\times}k_3^2)\\
\beta &= k_3^2 + h_1^2k_2^2
\end{aligned}$$

and the precomputed constants

$$d_1{=}f_0{+}\varepsilon h_1^4,\quad d_2{=}f_2{+}\varepsilon h_1^2{+}h_1^3,\quad c_1{=}\sqrt{d_1^2{+}h_1^2(f_0{+}h_1^2f_2)d_2},\quad c_2{=}\tfrac{d_1{+}h_1^2d_2{+}h_1^5}{h_1}\quad\text{and}\quad c_3{=}\tfrac{d_2}{h_1}$$

Assuming these constants (only depending on the curve) and the inverse of h_1^2 are precomputed, an addition requires 34 M and 2 S whereas a doubling

requires 21 M and 6 S. This is, of course, not satisfying but this is the worst case. Moreover, we observe that there are many multiplications by h_1 so it is interesting to choose a curve with $h_1 = 1$. In this case an addition can be done in 21M and 2S and a doubling in 13M and 6S which is much better.

5.2 Formulas for Curves of Type Ib

Remember that any curve of type Ib can be defined by an equation of the form

$$y^2 + (x^2 + h_1 x)y = x^5 + \varepsilon x^4 + f_2 x^2 + f_0.$$

Addition and doubling can be computed using the following formulas for φ_{ij} :

$$\begin{aligned}
\varphi_{11}(\mathcal{A},\mathcal{B}) &= (k_1{\times}l_4 + k_2{\times}l_3 + k_4{\times}l_1 + k_3{\times}l_2)^2 \\
\varphi_{12}(\mathcal{A},\mathcal{B}) &= (k_1 l_4 + k_4 l_1 + h_1{\times}k_1{\times}l_3 + h_1{\times}k_3{\times}l_1) \times (k_2 l_3 + k_3 l_2 + h_1 k_1 l_3 + h_1 k_3 l_1) \\
\varphi_{13}(\mathcal{A},\mathcal{B}) &= (k_4 l_1 + h_1 k_3 l_1){\times}\alpha + (k_1 l_4 + h_1 k_1 l_3){\times}\beta \\
\varphi_{14}(\mathcal{A},\mathcal{B}) &= \alpha{\times}\beta \\
\text{with } \alpha &= h_1{\times}(k_1 l_4 + k_2 l_3) + k_3{\times}l_3 \\
\beta &= h_1{\times}(k_4 l_1 + k_3 l_2) + k_3 l_3
\end{aligned}$$

$$\begin{aligned}
\varphi_{14}(\mathcal{A},\mathcal{A}) &= h_1^2{\times}\left(\left(k_1^2 + k_2^2\right) \times \left(k_3^2 + k_4^2\right) + k_1^2{\times}k_3^2 + k_2^2{\times}k_4^2\right) + k_3^4 \\
\varphi_{24}(\mathcal{A},\mathcal{A}) &= \alpha{\times}k_1^2 + h_1{\times}\left(k_2^2 k_4^2 + k_3^4 + d_1{\times}k_1^2 k_3^2\right) \\
\varphi_{34}(\mathcal{A},\mathcal{A}) &= h_1{\times}\alpha k_1^2 + h_1^2{\times}k_3^4 + k_3^2{\times}k_4^2, \\
\varphi_{44}(\mathcal{A},\mathcal{A}) &= h_1{\times}\left(h_1\alpha k_1^2 + k_3^2 k_4^2\right) + \left(k_4^2 + c_1{\times}k_1^2 + c_2{\times}k_2^2\right)^2 + c_3{\times}k_3^4 \\
\text{with } \alpha &= f_0 h_1{\times}\left(h_1^2{\times}k_1^2 + k_2^2 + \frac{1}{h_1^2}{\times}k_3^2\right)
\end{aligned}$$

with the precomputed constants

$$d_1 = f_2 + \varepsilon h_1^2 + h_1^3 + \frac{f_0}{h_1^2}, c_1 = \sqrt{f_0 h_1^2 d_1}, c_2 = \sqrt{f_0 h_1^2} \text{ and } c_3 = f_2 + \varepsilon h_1^2$$

Assuming these constants and $\frac{1}{h_1^2}$ are precomputed, an addition requires 18 M and 1 S whereas a doubling requires 17 M and 6 S. This is more satisfying than type Ia. We again observe that there are many multiplications by h_1. So, if $h_1 = 1$ an addition can be done in only 14M and 1S and a doubling in 10M and 6S. This becomes interesting and competitive with other methods. Indeed, the best known method for classical scalar multiplication algorithms requires 35M and 6S for doubling (33M and 6S if $h_1 = 1$) whereas our method requires 35M and 7S (24M and 7S if $h_1 = 1$) for both a doubling and an addition, so it is inevitably better. In fact, we can do even better with curves of type II.

5.3 Formulas for Curves of Type II

Remember that any curve of type II can be defined by an equation of the form

$$y^2 + xy = x^5 + f_3 x^3 + \varepsilon x^2 + f_0.$$

Addition and doubling can be computed using the following formulas for φ_{ij} :

$$\begin{aligned}
\varphi_{11}(\mathcal{A},\mathcal{B}) &= (\alpha+\beta)^2 \\
\varphi_{12}(\mathcal{A},\mathcal{B}) &= (k_1{\times}l_3 + k_3{\times}l_1)^2 \\
\varphi_{13}(\mathcal{A},\mathcal{B}) &= k_3 l_1{\times}\alpha + k_1 l_3{\times}\beta \\
\varphi_{14}(\mathcal{A},\mathcal{B}) &= \alpha{\times}\beta \\
\text{with } \alpha &= k_1{\times}l_4 + k_2{\times}l_3 \\
\beta &= k_4{\times}l_1 + k_3{\times}l_2
\end{aligned}$$

$$\begin{aligned}
\varphi_{14}(\mathcal{A},\mathcal{A}) &= k_1^2{\times}k_4^2 + k_2^2{\times}k_3^2 \\
\varphi_{24}(\mathcal{A},\mathcal{A}) &= k_1^2{\times}k_3^2 \\
\varphi_{34}(\mathcal{A},\mathcal{A}) &= \left(k_3^2 + \sqrt{f_0}{\times}k_1^2\right)^2 \\
\varphi_{44}(\mathcal{A},\mathcal{A}) &= \left(f_3{\times}\left(k_3^2 + \sqrt{f_0}k_1^2\right) + \sqrt{f_0}{\times}k_2^2 + k_4^2\right)^2
\end{aligned}$$

These formulas are very simple and easy to evaluate. There is no doubt that they provide very fast arithmetic on genus 2 curves in characteristic 2. Indeed, assuming $\sqrt{f_0}$ is precomputed, an addition requires 12 M and 2 S whereas a doubling requires 6 M and 6 S. This means that a complete scalar multiplication using Montgomery ladder requires only 18 M (and even 16 if $f_0 = 1$) and 8 S for each bit of the exponent whereas 20M and 8S are required for the doubling alone in "recent" projective coordinates ([19]). It is more delicate to compare this method with elliptic curve without a fully optimized implementation. However, our method seems to be competitive with Montgomery ladder on elliptic curves (18 multiplications in $\mathbb{F}_{2^d}$ against 6 in $\mathbb{F}_{2^{2d}}$). Let us now give more detailed comparisons.

5.4 Comparison With Usual Algorithms for Scalar Multiplication

To date, the best algorithms for scalar multiplication on genus 2 curves in characteristic 2 are obtained by using mixed projective coordinates or variants [20, 19]. In this case, Lange needs around 40 multiplications or squaring both for a mixed addition and for a doubling. However, we can use efficient algorithms (like the sliding window method) whereas, in Montgomery ladder, we must perform both an addition and a doubling for each bit of the exponent. Before comparing efficiencies more precisely, let us put advantages of Montgomery ladder forward.

– As was the case for elliptic curves, Montgomery ladder is naturally resistant to side-channel attacks, contrary to other algorithms for scalar multiplications. For this reason it is of great interest to people who need to implement

hyperelliptic curves protocols on smart cards or other systems which are sensitive to side-channel attacks.
– 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 6 or 8 of them so it is also interesting in terms of memory usage. This is an advantage for constrained environments.

In order to compare efficiency of our method with usual methods, we give in table 1, for each type of curve, complexities for our algorithm and for a sliding window method with window size 4 using the best system of coordinates. In practice, sizes 3 or 4 are used but, for objectivity, we choose to make our comparisons with a lower bound for complexities. The system of coordinates used is the so called "recent coordinate" for type II curves [19] and "new coordinate" for type I curves (except type Ia with $h_1 \neq 1$ where projective coordinates are more appropriate) [20]. These complexities are given for one bit of the exponent (on average for the sliding window method but we do not count the precomputations so the complexities given for sliding window method are underestimated). In order to obtain comparisons as objective as possible, we also give an equivalent in number of multiplication assuming that performing a square in $\mathbb{F}_{2^d}$ is either as expensive as 0.3 multiplications (which is usually the case in polynomial basis) or free (which is the case in normal basis). Finally we give the gain obtained using our method.

Type of the curve	Ia	Ia ($h_1 = 1$)	Ib	Ib ($h_1 = 1$)	II
Lange's formulas					
Double	38M+7S	33M+6S	37M+6S	33M+6S	20M+8S
Addition	38M+4S	35M+5S	37M+4S	35M+4S	42M+7S
Sliding window	45.6M+7.8S	40M+7S	44.4M+6.8S	40M+6.8S	28.4M+9.4S
with S=0.3M	48M	42.1M	46.5M	42M	31.2M
with S=0	45.6M	40M	44.4M	40M	28.4M
Kummer surface					
Double	21M+6S	13M+6S	17M+6S	10M+6S	6M+6S
Addition	34M+2S	21M+2S	18M+S	14M+S	12M+2S
Montgomery ladder	55M+8S	34M+8S	35M+7S	24M+7S	18M+8S
with S=0.3M	57.4M	36.4M	37.1M	26.1M	20.4M
with S=0	55M	34M	35M	24M	18M
gain with S=0.3M	-19.5%	13.5%	20%	38%	35%
gain with S=0	-20.5%	15%	21%	40%	37%

Table 1. Comparison between sliding window method and Montgomery ladder

Our new method is not interesting in term of efficiency for type Ia curves with $h_1 \neq 1$. In fact we obtain in this case a similar result as in odd charac-

teristic [6]: Montgomery ladder is less efficient than usual scalar multiplication algorithm but is still interesting because of the reasons shown at the beginning of this section (mainly side-channel attack resistance). Our method becomes very interesting in term of efficiency for curves of type Ib and II since gain between 20 and 40 percent are obtained. Let us note that for these types of curves, Montgomery ladder cannot be slower that any other scalar multiplication algorithm (with the systems of coordinates known to date of course) since for each bit of the exponent it requires less operations than a double alone.

With such results, it is interesting to compare hyperelliptic cryptosystems with elliptic ones.

5.5 Comparison with elliptic curves

The main interest of hyperelliptic cryptosystems is that, for the same level of security, the base field is chosen twice as small as the one used for elliptic cryptosystems. Thus base field operations are cheaper in hyperelliptic cryptosystems. More precisely, for cryptographic applications, the base field arithmetic uses either the school-book or the Karatsuba algorithm so the cost of a base field multiplication in elliptic cryptosystems is at least three times the cost in hyperelliptic cryptosystems. We assume in the following that this minimal ratio holds. However, it is quite delicate to do an objective comparison since this ratio is dependent on the device used (architecture, multiprecision algorithms, size of fields,...). For instance, in constraint environments, doubling the size of the base field can be very expensive and we can take more advantage of our new method in such a case.

For elliptic curves in characteristic 2, Montgomery ladder is the best known method to perform the scalar multiplication and it requires 6 M and 4 S which is equivalent, under our optimistic assumptions, to 18 M and 12 S in $\mathbb{F}_{2^d}$. We can see that this complexity is approximately the same than the one obtained for type II curves (18 M and and 8 S). This leads to a 5 percent gain if S=0.3M. It would be of course more appropriate to perform a real life comparison as the one presented by Bernstein at ECC 2006 in odd characteristic following the work of Gaudry [11]. At present, our team is achieving a library in order to be able to perform this type of comparisons [14] but it is a long term work. Anyway, we are confident that we can beat elliptic curves since Gaudry obtained similar results in odd characteristic using similar assumptions.

5.6 Comparison with Gaudry-Lubicz formulas

Recently, in [12], Gaudry announced he was able to generalize his previous work ([11]) to characteristic 2 and he is currently writing a paper with Lubicz on the subject. Their approach is different and uses theta functions to obtain formulas for doubling and pseudo-addition requiring only 7M and 5S for the doubling and 11M and 4S for the pseudo-addition. However these formulas can only be used

for curves of type Ib with $h_1 = 1$. In this case, Gaudry and Lubicz then obtain a Montgomery ladder which is 20 to 25 % facter than ours (depending on the cost of squarings in $\mathbb{F}_{2^d}$). Unfortunately, their method cannot be used for the other cases (for the moment).

6 Conclusion and Prospects

Thanks to the theory of the Kummer surface of a hyperelliptic curve of genus 2 in characteristic 2, we generalized to genus 2 curves the method of Montgomery for scalar multiplication on elliptic curves. Contrary to odd characteristic, the formulas obtained are competitive for all genus 2 curves. This is not so surprising since this was already the case for elliptic curves. All these formulas are available on [8] in magma format. For genus 2 curves of type Ia with $h_1 \neq 1$, our method is less efficient than usual scalar multiplication algorithms on genus 2 curves but has the advantage to be resistant to side-channel attacks. This was already the case in odd characteristic so this kind of result was expected. It is more surprising that Montgomery ladder for genus 2 curves provides gains of efficiency between 13 and 40 percent for curves of type Ia (with $h_1 = 1$), Ib and II compared to best algorithms known to date (which are not protected against side-channel attack). On the contrary, our method is less efficient than Gaudry-Lubicz one for curves of type Ib with $h_1 = 1$. Finally, under some reasonable assumptions, we also show that type II curves can beat elliptic curves. Hence, the Montgomery ladder for genus 2 curves can be of interest even for people who are not working on embedded devices.

Finally we provide an algorithm for scalar multiplication which is not only resistant to side-channel attack but is also very efficient. This proves, if needed, that hyperelliptic curve cryptosystems are a good alternative to elliptic ones. Of course, we strongly recommend to use curves of type II for efficient implementations of hyperelliptic curves cryptosystem in characteristic 2.

References

1. Brier, E., Joye, M.: Weierstrass Elliptic Curves and Side-Channel Attacks, Public Key Cryptography, Lecture Notes in Computer Science, **2274** (2002)
2. Byramjee, B., Duquesne, S.: Classification of genus 2 curves over $\mathbb{F}_{2^n}$ and optimization of their arithmetic, Cryptology ePrint Archive 107 (2004)
3. Cantor, D. G.: Computing on the Jacobian of a hyperelliptic curve. Math. Comp., **48** (1987) 95–101
4. Choie, Y., Yun, D.: Isomorphism classes of hyperelliptic curves of genus 2 over $\mathbb{F}_q$, ACISP 2002, Lecture Notes in Computer Science, **2384** (2002) 190–202
5. Cohen, H., Frey, G.: Handbook of elliptic and hyperelliptic curve cryptography, Discrete Math. Appl., Chapman & Hall/CRC (2006)
6. Duquesne, S.: Montgomery scalar multiplication for genus 2 curves, ANTS-VI, Lecture Notes in Computer Science, **3076** (2004) 153–168

7. Duquesne, S.: Traces of the group law on the Kummer surface of a curve of genus 2 in characteristic 2, preprint, available at [8]
8. Duquesne, S.: Formulas for traces of the group law on the Kummer surface of a curve of genus 2 in characteristic 2, available at `http://www.math.univ-montp2.fr/~duquesne/articles/kummer2`
9. Galbraith, S.: Supersingular curves in cryptography, Asiacrypt 2001, Lecture Notes in Computer Science, **2248** (2001) 495–513
10. Gaudry, P., Hess, F., Smart, N.: Constructive and destructive facets of Weil descent on elliptic curves, J. Cryptology **15:1** (2002) 19-46
11. Gaudry, P.: Fast genus 2 arithmetic based on Theta functions, Journal of Mathematical Cryptology **1** (2007) 243–265
12. Gaudry, P.: Variants of the Montgomery form based on Theta functions, November 2006, Toronto and preprint with D. Lubicz (2008)
13. Harley, R.: Fast arithmetic on genus 2 curves, available at http://cristal.inria.fr/∼harley/hyper (2000)
14. Imbert, L., Peirera, A., Tisserand, A.: A Library for Prototyping the Computer Arithmetic Level in Elliptic Curve Cryptography, Proc. SPIE, **6697** (2007) 66970N
15. Koblitz, N.: Elliptic curve cryptosystems, Math. Comp., **48** (1987) 203–209
16. Koblitz, N.: Algebraic aspects of cryptography, Algorithms and Computation in Mathematics, **3** (1998)
17. 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
18. Kocher, P. C., Jaffe, J., Jun, B.: Differential power analysis, CRYPTO'99, Lecture Notes in Computer Science, **1666** (1999) 388-397
19. Lange, T.: to appear
20. Lange, T.: Formulae for arithmetic on genus 2 hyperelliptic curves, Appl. Algebra Engrg. Comm. Comput. **15:5** (2005) 295-328
21. Lopez, J., Dahab, R.: Improved algorithms for elliptic curve arithmetic in GF(2^n), Tech. Report IC-98-39, Relatrio Tcnico, October 1998
22. Lopez, J., Dahab, R.: Fast multiplication on elliptic curves over GF(2^m) without precomputation, CHES 1999, Lecture Notes in Computer Science, **1717** (1999) 316-327
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, 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
27. 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
28. Smart, N., Siksek, S.: A fast Diffe-Hellman protocol in genus 2, Journal of Cryptology, **12** (1999) 67–73