Another approach to pairing computation in Edwards coordinates

Sorina Ionica[2] and Antoine Joux[1,2]

[1] Université de Versailles Saint-Quentin-en-Yvelines, 45 avenue des États-Unis, 78035 Versailles CEDEX, France
[2] DGA
sorina.ionica,antoine.joux@m4x.org

Abstract. The recent introduction of Edwards curves has significantly reduced the cost of addition on elliptic curves. This paper presents new explicit formulae for pairing implementation in Edwards coordinates. We prove our method gives performances similar to those of Miller's algorithm in Jacobian coordinates and is thus of cryptographic interest when one chooses Edwards curve implementations of protocols in elliptic curve cryptography. The method is faster than the recent proposal of Das and Sarkar for computing pairings on supersingular curves using Edwards coordinates.

Keywords: Tate pairing, Miller's algorithm, Edwards coordinates.

1 Introduction

Pairings on elliptic curves are currently of great interest due to their applications in a number of cryptographic protocols such as the tripartite Diffie-Hellman protocol [20], identity-based encryption [6], short signatures [7] and group signatures [8]. In this paper we propose to reassess the computational cost of pairings in the light of the introduction by Edwards [14] of a new representation of the addition law on elliptic curves. Recently, a method for computing pairings in Edwards coordinates for supersingular curves was proposed in [13]. The approach proposed in the present paper is very different from [13].

Our starting point concerning Edwards curves is a generalized result of Bernstein and Lange [3]. They showed that an elliptic curve defined over a field K of characteristic different from 2 is birationally equivalent over some extension of K to an Edwards curve, i.e. a curve of the form $x^2 + y^2 = 1 + dx^2y^2$ with $d \notin \{0, 1\}$. A simple and symmetric addition law can be defined on such a curve:

$$(x_1, y_1), (x_2, y_2) \to \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2} \right). \quad (1)$$

Bernstein and Lange showed that this addition law is, in fact, the standard addition law on the corresponding elliptic curve and gave explicit formulae for additions and doublings, which are faster than all previously known formulae.

The basic algorithm used in pairing computation was first described by Miller and is an extension of the double-and-add method for finding a point multiple. Our goal in this paper is to extend Miller's algorithm to allow computation of pairings on curves given in Edwards coordinates. The difficulty when trying to express Miller's algorithm in Edwards coordinates is that it is hard to find the equations of rational functions that need to be evaluated at each addition step. On a curve in Weierstrass form, these equations correspond to straight lines. For curves in Edwards form matters are more complex.

Our main idea is to describe a map of degree 4 from the Edwards curve to a curve $E_{s,p} : s^2p = (1 + dp)^2 - 4p$. This curve has an equation of total degree 3 and, as in the Weierstrass case, we can easily compute the equations of the two lines that appear naturally when adding two points P_1 and P_2, i.e. the line l passing through P_1 and P_2 and the vertical line v that passes through $P_1 + P_2$. We then pullback l and v to the Edwards curve. The output of our algorithm is essentially the desired pairing. More precisely, we obtain the 4-th power of the usual pairing.

The remainder of this paper is organised as follows: Section 2 recalls basic properties of Edwards curves and of the Edwards addition law. It also presents Miller's algorithm on an elliptic curve given by a Weierstrass equation. Section 3 introduces the curve $E_{s,p}$ and explains how to compute pairings on Edwards curves by using this representation. Finally, in section 4 we give estimates of the computational cost of the Tate pairing in Edwards coordinates and compare this cost to that of a pairing implementation in Jacobian coordinates (for a Weierstrass equation). We only treat the case of curves with even embedding degree, which is prefered in most of the cryptographic applications. We compare the efficiency of our suggestion to the use of Jacobian coordinates, which is, to the best of our knowledge, the fastest existing method for computing pairings. A proposal for the operation count in Jacobian coordinates using recent formulas from [2] is presented in Appendix A. In the case of supersingular curves, we also compare our method to results obtained for supersingular curves in [13].

2 Preliminaries

2.1 Edwards coordinates

Edwards showed in [14] that every elliptic curve E defined over an algebraic number field is birationally equivalent over some extension of that field to a curve given by the equation:

$$x^2 + y^2 = c^2(1 + x^2y^2). \tag{2}$$

In this paper, we make use of the results concerning elliptic curves over finite fields obtained by Bernstein et al. [1]:

Theorem 1. Fix a finite field $\mathbb{F}_q$ with $\mathrm{char}(\mathbb{F}_q) \neq 2$. Let E be an elliptic curve over $\mathbb{F}_q$. E is birationally equivalent over $\mathbb{F}_q$ to a curve $x^2 + y^2 = 1 + dx^2y^2$ if and only if the group $E(\mathbb{F}_q)$ has an element of order 4.

In the sequel, we call the curve $x^2+y^2 = 1+dx^2y^2$ an Edwards curve. It was shown in [3] that an Edwards curve E is birationally equivalent to the elliptic curve $E_d : (1/(1-d))v^2 = u^3 + 2((1+d)/(1-d))u^2 + u$ via the rational map:

$$\begin{aligned} \psi : \quad E_d &\to E \\ (u,v) &\to \left(\frac{2u}{v}, \frac{(u-1)}{(u+1)}\right). \end{aligned} \tag{3}$$

On an Edwards curve, we consider the following addition law:

$$(x_1,y_1),(x_2,y_2) \to \left(\frac{x_1y_2+y_1x_2}{1+dx_1x_2y_1y_2}, \frac{y_1y_2-x_1x_2}{1-dx_1x_2y_1y_2}\right). \tag{4}$$

The neutral element of this addition law is $O = (0,1)$. For every point $P = (x,y)$ the opposite element is $-P = (-x,y)$. The curve has a 4-torsion subgroup defined over $\mathbb{F}_q$. We note $T_2 = (0,-1)$ the point of order 2 and $T_4 = (1,0)$, $-T_4 = (-1,0)$ the two points of order 4.

In [3], it was shown that this addition law is *complete* when d is not a square. This means it is defined for all pairs of input points on the Edwards curve with no exceptions for doublings, neutral element etc. Moreover, this addition law is the same as the one induced by the birational map described above, i.e. $P_1 + P_2 = \psi^{-1}(P_1) + \psi^{-1}(P_2)$, where the last $+$ stands for the standard addition law on the elliptic curve E_d. Note that ψ extends to exceptional points: $\psi(O_{E_d}) = (0,1)$, where O_{E_d} is the neutral element of E_d, $\psi((0,0)) = (0,-1)$. See [1] for a complete description on exceptional points.

In the following sections we use projective coordinates. A projective point (X,Y,Z) satisfying $(X^2+Y^2)Z^2 = Z^4 + dX^2Y^2$ and $Z \neq 0$ corresponds to the affine point $(X/Z, Y/Z)$ on the curve $x^2+y^2 = 1+dx^2y^2$. The Edwards curve has two points at infinity $(0:1:0)$ and $(1:0:0)$. These points are actually singularities of the curve and, as stated in [3], resolving them produces four points defined over $\mathbb{F}_q(\sqrt{d})$. If d is not a square in $\mathbb{F}_q$ then this is a quadratic extension of $\mathbb{F}_q$.

Edwards curves became interesting for elliptic curve cryptography when it was proven by Bernstein and Lange in [3] that they provide addition and doubling formulae faster than all addition formulae known at that time. In the sequel we also make use of inverted Edwards coordinates [4]. A point (X,Y,Z) in *inverted Edwards coordinates* stands for the point $(Z/X, Z/Y)$ on the affine Edwards curve. Table 1 below gives a cost comparison between operations of addition, doubling and mixed addition (i.e. the Z-coordinate of one of the two points is 1) on the Edwards curve and on the Weierstrass form in Jacobian coordinates. We briefly remind the reader that a point (X,Y,Z) in Jacobian coordinates corresponds to the affine point (x,y) with $x = X/Z^2$ and $y = Y/Z^3$. We denote by $\mathbf{M}$ the cost of a field multiplication and by $\mathbf{S}$ the cost of a field squaring. We assume that the cost of addition and that of multiplication by d are negligible (we choose d a small constant). Results are taken from [2].

Table 1. Performance evaluation: Edwards versus Jacobian

	Edwards coordinates	inverted Edwards coordinates	Jacobian coordinates
addition	10**M**+1**S**	9**M**+1**S**	11**M**+5**S**
doubling	3**M**+4**S**	3**M**+4**S**	1**M**+8**S** or 3**M**+5**S** for $a = -3$
mixed addition	9**M**+1**S**	8**M**+1**S**	7**M**+4**S**

2.2 Background on pairings

In this section we give a brief overview of the definition of the Tate pairing and of Miller's algorithm [22] used in pairing computations. This algorithm heavily relies on the double and add method for finding a point multiple. Let E be an elliptic curve given by a Weierstrass equation:

$$y^2 = x^3 + ax + b, \tag{5}$$

defined over a finite field $\mathbb{F}_q$. Consider r a large prime dividing $\#E(\mathbb{F}_q)$ and k the corresponding embedding degree, i.e. the smallest positive integer such that r divides $q^k - 1$. Let P_∞ denote the neutral element on the elliptic curve.

Let P be an r-torsion point and for any integer i, denote by $f_{i,P}$ the function with divisor $\mathrm{div}\,(f_{i,P}) = i(P) - (iP) - (i-1)(P_\infty)$ (see [23] for an introduction to divisors). Note $f_{r,P}$ is such that $\mathrm{div}\,(f_{r,P}) = r(P) - r(P_\infty)$.

In order to define the Tate pairing we take Q an element of $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$. Let T be a point on the curve such that the support of the divisor $D = (Q + T) - (T)$ is disjoint from the one of $f_{r,P}$. We then define the Tate pairing as:

$$t_r(P, Q) = f_{r,P}(D). \tag{6}$$

This value is a representative of an element of $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^r$. However for cryptographic protocols it is essential to have a unique representative so we will raise it to the $((q^k - 1)/r)$-th power, obtaining an r-root of unity. We call the resulting value the *reduced* Tate pairing:

$$T_r(P, Q) = t_r(P, Q)^{\frac{q^k-1}{r}}.$$

As stated in [16] if the function $f_{r,P}$ is normalized, i.e. $(u_0^r f_{r,P})(P_\infty) = 1$ for some $\mathbb{F}_q$-rational uniformizer u_O at P_∞, then one can ignore the point T and compute the pairing as:

$$T_r(P, Q) = f_{r,P}(Q)^{(q^k-1)/r}.$$

In the sequel of this paper we only consider normalized functions. Before going into the details of Miller's algorithm, we recall the standard addition law on an elliptic curve in Weierstrass form. Suppose we want to compute the sum of iP and jP for $i, j \geq 1$. Let l be the line through iP and jP. Then l intersects the cubic curve E at one further point R according to Bezout's theorem (see [18]).

We take v the line between R and P_∞ (which is a vertical line when R is not P_∞). Then v intersects E at one more point which is defined to be the sum of iP and jP, that is $(i+j)P$.

The lines l and v are functions on the curve and the corresponding divisors are:

$$\begin{aligned}\operatorname{div}(l) &= (iP) + (jP) + (R) - 3(P_\infty),\\ \operatorname{div}(v) &= (R) + ((i+j)P) - 2(P_\infty).\end{aligned}$$

One can then easily check the following relation:

$$f_{i+j,P} = f_{i,P} f_{j,P} \frac{l}{v}. \tag{7}$$

In the sequel, we will call this relation *Miller's equation*. Turning back to Miller's algorithm, suppose we want to compute $f_{r,P}(D)$. We compute at each step of the algorithm on one side mP, where m is the integer with binary expansion given by the i topmost bits of the binary expansion of r, and on the other side $f_{m,P}$ evaluated at D, by exploiting the formula above. We call the set of operations executed for each bit i of r a *Miller operation*.

Algorithm 1 Miller's algorithm

INPUT: An elliptic curve E defined over a finite field $\mathbb{F}_q$, P an r-torsion point on the curve and $Q \in E(\mathbb{F}_{q^k})$.
OUTPUT: the Tate pairing $t_r(P,Q)$.
Let $i = [\log_2(r)]$, $K \leftarrow P$,$f \leftarrow 1$.
while $i \geq 1$ **do**
  Compute equations of l and v arising in the doubling of K.
  $K \leftarrow 2K$ and $f \leftarrow f^2 l(Q)/v(Q)$.
  if the i-th bit of r is 1 **then**
    Compute equations of l and v arising in the addition of K and P.
    $K \leftarrow P + K$ and $f \leftarrow f l(Q)/v(Q)$.
  end if
  Let $i \leftarrow i - 1$.
end while

The advantage of dealing with the Weierstrass form when running the algorithm is that the equations of l and v are easy to find as they already appear in the addition process. This is obviously not the case with the Edwards curve, whose equation has degree 4. It is difficult to describe the equation of a function with divisor equal to $\operatorname{div}(f_{i+j,P}/f_{i,P}f_{j,P})$ and to establish a relation of type (7). An idea would be to consider Miller's equation on the birationally equivalent Weierstrass curve and then transport this equation on the Edwards curve. However this yields an unefficient pairing computation. Our proposal is to map the Edwards curve to another genus 1 curve with an equation of degree 3, get l and v as straight lines and then pull them back to the Edwards curve.

3 Pairings on Edwards curves

In this section, E denotes an Edwards curve defined over some finite field $\mathbb{F}$ of odd characteristic. Let us take a look at the action of the 4-torsion subgroup defined over $\mathbb{F}$ on a fixed point on the Edwards curve $P = (x, y)$, with $xy \neq 0$. A simple computation shows that $P + T_4 = (y, -x)$, $P + T_2 = (-x, -y)$ and $P - T_4 = (-y, x)$. We notice then that by letting $p = (xy)^2$ and $s = x/y - y/x$, the pair (p, s) characterizes the point P up to an addition with a 4-torsion point. This leads us to consider the following morphism from the Edwards curve to a curve given by the equation $E_{s,p} : s^2 p = (1 + dp)^2 - 4p$

$$\begin{aligned} \phi : E &\to E_{s,p} \\ (x, y) &\to ((xy)^2, \frac{x}{y} - \frac{y}{x}). \end{aligned}$$

In this section we study the arithmetic of the curve $E_{s,p}$, establish Miller's equation on this curve and then take its pullback, getting Miller's equation, this time on the Edwards curve. This yields all the tools needed to apply Miller's algorithm on the Edwards curve.

3.1 Arithmetic of the curve $s^2p = (1 + dp)^2 - 4p$

In this section we study the arithmetic of the curve:

$$E_{s,p} : s^2 p = (1 + dp)^2 - 4p.$$

The equation of $E_{s,p}$ in homogeneous coordinates (P, S, Z) is given by $S^2 P = (Z + dP)^2 Z - 4PZ^2$. If we dehomogenize this equation by putting $P = 1$ we get the Weierstrass equation of an elliptic curve:

$$s^2 = z^3 + (2d - 4)z^2 + d^2 z. \tag{8}$$

We note $O_{s,p} = (0, 1, 0)$ the point at infinity and $T_{2,s,p} = (1, 0, 0)$ which is a two torsion point. The following definition is simply another way to write the addition law on an elliptic curve in (p, s) coordinates.

Definition 1. *Let $P_1, P_2 \in E_{s,p}$, L the line connecting P_1 and P_2 (tangent line to $E_{s,p}$ if $P_1 = P_2$), and R the third point of intersection of L with E. Let L' be the vertical line through R (of equation $p = p_R$). Then $P_1 + P_2$ is the point such that L' intersects $E_{s,p}$ at R and $P_1 + P_2$ (the point symmetric to R with respect to the p-axis).*

Note that we can extend ϕ to the 4-torsion points by $\phi(O) = \phi(T_2) = \phi(T_4) = \phi(-T_4) = O_{s,p}$.

Theorem 2. *Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points on the Edwards curve and P_3 their sum. Then $\phi(P_3)$ is the sum of $\phi(P_1)$ and $\phi(P_2)$ in the addition law of Definition 1.*

Proof. Consider $\psi : E_d \to E$ the map defined in equation (3). By using Proposition 2.1 in [23] one can easily see that $\phi \circ \psi$ is a morphism from E_d to the elliptic curve $E_{s,p}$. As $\phi \circ \psi(O_{E_d}) = O_{s,p}$ (where O_{E_d} is the point at infinity of E_d), we deduce that $\phi \circ \psi$ is an isogeny. Moreover it was shown in Theorem 3.2 of [3] that the Edwards addition law on E is the same as the addition law induced by ψ. It follows that the addition law induced by ϕ is the same as the standard addition law on the elliptic curve, so it corresponds to the addition law described at Definition 1. □

As in the sequel we need to compute the pullback of certain functions on the curve $E_{s,p}$ we now compute the degree of this map.

Proposition 1. *The map $\phi : E \to E_{s,p}$ is separable of degree 4.*

Proof. Let $P = (x, y)$ be a point on the Edwards curve. The doubling formula gives:

$$2P = \left(\frac{2xy}{1 + d(xy)^2}, \frac{y^2 - x^2}{1 - d(xy)^2}\right) = \left(\frac{2xy}{x^2 + y^2}, \frac{y^2 - x^2}{2 - (x^2 + y^2)}\right).$$

If $xy \neq 0$ then by letting $p = (xy)^2$ and $s = x/y - y/x$ we can write:

$$4P = \left(\frac{4ps(1 - d^2p^2)}{(1 - d^2p^2)^2 - 4dp^2s^2}, \frac{4p(1 + dp)^2 - ps^2}{(1 - d^2p^2)^2 + 4dp^2s^2}\right).$$

This means that by defining:

$$\begin{aligned} \theta : \quad & E_{s,p} \to E \\ & (p, s) \to \left(\frac{4ps(1 - d^2p^2)}{(1 - d^2p^2)^2 - 4dp^2s^2}, \frac{4p(1 + dp)^2 - ps^2}{(1 - d^2p^2)^2 + 4dp^2s^2}\right), \end{aligned}$$

we get a rational map ψ such that $\phi \circ \theta = [4]$ on E. It follows that $\deg \phi$ divides 16. As the inseparable degree $\deg_i \phi$ is a power of the characteristic of $\mathbb{F}_q$, we deduce that ϕ is a separable map (we have supposed that $\mathrm{char}(\mathbb{F}_q) \neq 2$). By putting $\phi(P) = Q$ we easily get $\phi^{-1}(Q) = \{P, P + T_2, P + T_4, P - T_4\}$. We conclude that $\deg \phi = 4$.□

3.2 Miller's algorithm on the Edwards curve

Let P be an r-torsion point on the Edwards curve. We consider slightly modified functions $f^{(4)}_{i,P}$:

$$\begin{aligned} f^{(4)}_{i,P} = i((P) + (P + T_4) + (P + T_2) + (P - T_4)) - ((iP) + (iP + T_4) \\ + (iP + T_2) + (iP - T_4)) - (i - 1)((O) + (T_4) + (T_2) + (-T_4)). \end{aligned}$$

Then $f^{(4)}_{r,P} = r((P)+(P+T_4)+(P+T_2)+(P-T_4))-r((O)+(T_4)+(T_2)+(-T_4))$, which means that we can compute the Tate pairing up to a 4-th power:

$$T_r(P, Q)^4 = f^{(4)}_{r,P}(Q)^{\frac{q^k - 1}{r}}.$$

We also get the following Miller equation:

$$f^{(4)}_{i+j,P} = f^{(4)}_{i,P} f^{(4)}_{j,P} \frac{l}{v}, \tag{9}$$

where l/v is the function of divisor:

$$\begin{aligned}\text{div}\,(l/v) = &((iP) + (iP + T_4) + (iP + T_2) + (iP - T_4))\\ &+((jP) + (jP + T_4) + (jP + T_2) + (jP - T_4))\\ &-(((i+j)P) + ((i+j)P + T_4) + ((i+j)P + T_2) + ((i+j)P - T_4)))\\ &-((O) + (T_4) + (T_2) + (-T_4)).\end{aligned}$$

Let $P' = \phi(P)$ and let $l_{s,p}$ and $v_{s,p}$ be functions on the $E_{s,p}$ curve such that div $(l_{s,p}) = (iP') + (jP') + (-(i+j)P') - 2(T_{2,s,p}) - (O_{s,p})$ and div $(v_{s,p}) = ((i+j)P') + (-(i+j)P') - 2(T_{2,s,p})$.

We observe that we have $l/v = \phi^*(l_{s,p}/v_{s,p})$ up to constants in $\mathbb{F}_q$. It is easy to find the equations of $l_{s,p}$ and $v_{s,p}$ as they appear naturally in the definition of the sum $iP' + jP'$, namely $l_{s,p}$ is the line connecting iP' and jP', and $v_{s,p}$ is the vertical line through $(i+j)P'$. As we will see in the next section, we can compute their pullback via the map ϕ without any significant computational cost.

4 Pairing computation in Edwards coordinates

In this section, we take a look into the details of the computation of pairings in Edwards coordinates and give estimates of the computational costs of the Miller operation. We start by estimating the cost of evaluating the function $f^{(4)}_{r,P}(Q)$ in terms of the cost of the doubling part of a Miller operation, which is executed for every bit of r. This seems reasonable, as it gives an evaluation which is independent from any fast exponentiation techniques that might be used in the implementation of the algorithm, such as the sliding window method or the use of a signed Hamming weight representation for r. We recall that a signed representation $(m_{n-1}...m_0)_s$ is said to be in *non-adjacent form*, or NAF for short, if $m_i m_{i+1} = 0$, with $m_i \in \{-1, 0, 1\}$ for all $i \geq 0$. The advantage of using such a representation is that on average the number of non-zero terms in a NAF expansion of length n is $n/3$ (see [9] for a precise analysis of the NAF density).

Moreover, in many cryptographic applications it is possible to choose r with low Hamming weight. The construction of Cocks and Pinch as described in [5, p. 210] allows for r to be chosen arbitrarily, so a prime of low Hamming weight can be chosen. Further examples are provided by a construction of Brezing and Weng [10] for prime embeddings degrees k, extended in [15] for all odd $k < 200$. Note that if the loop length parameter r does not have low Hamming weight, it is sufficient to have some multiple of r whose Hamming weight is small (usually the elliptic curve group order $\sharp E(F_q)$).

Example 1. The following example is given in [13]. Consider $E : y^2 = x^3 + x$ over $\mathbb{F}_q$, with $q \equiv 3 \bmod 4$. This curve is supersingular and its corresponding Edwards form is $x^2 + y^2 = 1 - (xy)^2$, so $d = -1$. One may choose for instance $q = 2^{520} + 2^{363} - 2^{360} - 1$, $r = 2^{160} + 2^3 - 1$ or $q = 2^{1582} + 2^{1551} - 2^{1326} - 1, r = 2^{256} + 2^{225} - 1$.

During the past few years, research in efficient implementation of pairings focussed on the reduction of the loop length in Miller's algorithm. It was proven that for curves with a small Frobenius trace t, the parameter r giving the length of the loop can be replaced by t. The interested reader should refer to [19] for a discussion on the choice of the loop length parameter.

Therefore, in order to give a complete evaluation of the complexity, we also count the number of operations in the mixed addition step of the Miller operation and compare it to the mixed addition step in Jacobian coordinates. The reader may refer to [11] for global estimates of pairing computation in Jacobian coordinates for some families of curves with $k = 2$ (in particular those in Example 1). In a general case, estimates for the doubling part for the Weiestrass form can be found in [21] and in [17], but we improved these results using recent doubling formulae from [2] (see Appendix A). estimates of the cost of the mixed addition step for Jacobian coordinates are taken from [12].

The remainder of this paper presents efficient computation of the Tate pairing in Edwards coordinates for curves with even embedding degree. These curves are preferred in cryptographic applications because a major part of the computations is performed in a proper subfield of $\mathbb{F}_{q^k}$. However, the results obtained in Section 3 are independent of the embedding degree, so similar computations can be done in a general case.

4.1 The case of an even embedding degree

Koblitz and Menezes showed in [21] that if q and k are chosen such as $q \equiv 1 \pmod{12}$ and $k = 2^i 3^j$, then the arithmetic of the extension field $\mathbb{F}_{q^k}$ can be implemented very efficiently as this field can be built up as a tower of extension fields:

$$\mathbb{F}_q \subset \mathbb{F}_{q^{d_1}} \subset \mathbb{F}_{q^{d_2}} ... \subset \mathbb{F}_{q^k},$$

where the ith field $F_{q^{d_i}}$ is obtained by adjoining a root of some irreducible polynomial $X^{d_i/d_{i-1}} - \beta_i$ and $d_i/d_{i-1} \in \{2, 3\}$.

We denote by $\mathbf{m}, \mathbf{M}$ (respectively $\mathbf{s}, \mathbf{S}$) the costs of multiplications (respectively squarings) in the field $\mathbb{F}_q$ and in the extension $\mathbb{F}_{q^k}$. Then according to [21] we get

$$\mathbf{M} \approx v(k)\mathbf{m} \text{ and } \mathbf{S} \approx v(k)\mathbf{s},$$

where $v(k) = 3^i 5^j$. Moreover, a multiplication of an element in $\mathbb{F}_{q^k}$ by an element in $\mathbb{F}_q$ costs $k\mathbf{m}$ operations.

For efficiency reasons, we restrict the domain of the Tate pairing to a product of cyclic subgroups of order r on the elliptic curve. In general, the point P can be chosen such that $\langle P \rangle$ is the unique subgroup of order r in $E(\mathbb{F}_q)$. In order

to get a non-degenerate pairing, we take Q a point of order r in $E(\mathbb{F}_{p^k})\backslash E(\mathbb{F}_p)$. Moreover, if the embedding degree is even, it was shown that the subgroup $\langle Q\rangle \subset E(\mathbb{F}_{q^k})$ can be taken so that the x-coordinates of all its points lie in $\mathbb{F}_{q^{k/2}}$ and the y-coordinates are products of elements of $\mathbb{F}_{q^{k/2}}$ with $\sqrt{\beta}$, where β is a nonsquare in $\mathbb{F}_{q^{k/2}}$ and $\sqrt{\beta}$ is a fixed squareroot in $\mathbb{F}_{q^k}$ (see [21] for details).

The same kind of considerations apply to Edwards curves. We recall that the curve $E : x^2 + y^2 = 1 + dx^2y^2$ is birationally equivalent to the curve E_d, via the rational map $\psi : E_d \to E$. We choose $P^{'}$ and $Q^{'}$ on the E_d curve as explained above and then take $P = \psi(P^{'})$ and $Q = \psi(Q^{'})$. It follows that the coordinates of elements of $\langle P\rangle$ are in $\mathbb{F}_q$. The subgroup $\langle Q\rangle \in \mathbb{F}_{q^k}$ is such that its elements have y-coordinates in the quadratic subextension $\mathbb{F}_{q^{k/2}}$ and x-coordinates that can be written as products of elements of $\mathbb{F}_{q^{k/2}}$ with $\sqrt{\beta}$, for some element β of $\mathbb{F}_{q^{k/2}}$.

Doubling step We now take a look into the details of the computation of a Miller iteration. We note $K = (X_1, Y_1, Z_1)$. Following [3] the doubling formulas for $2K = (X_3, Y_3, Z_3)$ are:

$$\begin{aligned} X_3 &= 2X_1Y_1(2Z_1^2 - (X_1^2 + Y_1^2)),\\ Y_3 &= (X_1^2 + Y_1^2)(Y_1^2 - X_1^2),\\ Z_3 &= (X_1^2 + Y_1^2)(2Z_1^2 - (X_1^2 + Y_1^2)). \end{aligned}$$

On the curve $E_{s,p}$ we consider $l_{s,p}$ the tangent line to the curve at $\phi(K) = (p_1, s_1)$ and $v_{s,p}$ the vertical line passing through $\phi(2K) = (p_3, s_3)$. These lines have the following equations:

$$\begin{aligned} l_{s,p}(s,p) &= 2p_1^2s_1(s - s_1) - p_1(2d(1 + dp_1) - (s_1^2 + 4))(p - p_1),\\ v_{s,p}(s,p) &= p - p_3. \end{aligned}$$

Using the equation of the curve $E_{s,p}$ and then the expressions for s and p we get

$$\begin{aligned} l_{s,p} &= 2p_1^2s_1(s - s_1) - (2d(1 + dp_1)p_1 - (1 + dp_1)^2)\\ &= 2p_1^2s_1(s - s_1) + (1 - dp_1)(1 + dp_1)\\ &= (x_1y_1)^2(x_1^2 - y_1^2)(2x_1y_1(x/y - y/x) - (x_1^2 - y_1^2))\\ &+(2 - x_1^2 - y_1^2)(x_1^2 + y_1^2)((xy)^2 - (x_1y_1)^2). \end{aligned}$$

Consequently, making use of the Edwards curve equation, we get the following equations of normalized functions l and v in projective coordinates:

$$\begin{aligned} l(x,y) = l_1(x,y)/l_2 = &((X_1^2 + Y_1^2 - Z_1^2)(X_1^2 - Y_1^2)(2X_1Y_1(x/y - y/x)\\ &-2(X_1^2 - Y_1^2)) + Z_3(dZ_1^2(xy)^2 - (X_1^2 + Y_1^2 - Z_1^2)))/\\ &(2X_1Y_1(X_1^2 + Y_1^2 - Z_1^2)(X_1^2 - Y_1^2)),\\ v(x,y) = v_1(x,y)/v_2 = &(dZ_3^2(xy)^2 - (X_3^2 + Y_3^2 - Z_3^2))/(X_3^2 + Y_3^2 - Z_3^2). \end{aligned}$$

We now show that the computational cost of the doubling part in Miller's algorithm is significantly lower because we can ignore terms that lie in a proper subfield of $\mathbb{F}_{q^k}$. These terms can be ignored because k is the multiplicative order of q modulo r, so $(q^k-1)/r$ is a multiple of $q^{k'}-1$ for one proper divisor k' of k. So we ignore l_2 and v_2 because they depend only of the coordinates of P, so they lie in $\mathbb{F}_q$. Since $(xy)^2 \in \mathbb{F}_{q^{k/2}}$ and hence $v_1(Q) \in \mathbb{F}_{q^{k/2}}$, it follows that we can also ignore $v_1(Q)$. Hence the function evaluation step in the doubling part of Miller's algorithm becomes:

$$f^{(4)} \leftarrow (f^{(4)})^2 l_1(Q). \tag{10}$$

Note that multiplications by $(xy)^2$ and $x/y - y/x$ cost $(k/2)\mathbf{m}$ each ($x/y - y/x$ is the product of some element in $\mathbb{F}_{q^{k/2}}$ with $\sqrt{\beta}$). Also note that computing $x/y - y/x$ costs one inversion in $\mathbb{F}_{q^{k/2}}$. In some protocols Q is a fixed point, so we can precompute $x/y - y/x$.

If $k = 2$, we actually have $(xy)^2 \in \mathbb{F}_q$, so we compute:

$$\begin{aligned} l_1(x,y) = ((X_1^2 + Y_1^2 - Z_1^2)(X_1^2 - Y_1^2)) \cdot 2X_1Y_1(x/y - y/x) - ((X_1^2 + Y_1^2 - Z_1^2) \\ \cdot(X_1^2 - Y_1^2)) \cdot 2(X_1^2 - Y_1^2) - Z_3 \cdot (dZ_1^2 \cdot (xy)^2 - (X_1^2 + Y_1^2 - Z_1^2)), \end{aligned}$$

For $k > 2$ some operations are done in $\mathbb{F}_{q^k}$ and others in $\mathbb{F}_q$, so we compute l_1 as it follows:

$$\begin{aligned} l_1(x,y) = ((X_1^2 + Y_1^2 - Z_1^2)(X_1^2 - Y_1^2)) \cdot 2X_1Y_1(x/y - y/x) - ((X_1^2 + Y_1^2 - Z_1^2) \\ \cdot(X_1^2 - Y_1^2)) \cdot 2(X_1^2 - Y_1^2) - Z_3 \cdot dZ_1^2 \cdot (xy)^2 + Z_3 \cdot (X_1^2 + Y_1^2 - Z_1^2), \end{aligned}$$

As an example, we detail the computation of the doubling step for $k > 2$ in Table 2. A Pari-Gp script proving correctness of this computation is given in Appendix B. Results are summarized in Table 3. The computations in the general

Table 2. Operations of the doubling part of the Miller operation for $k > 2$

$A \leftarrow X_1^2,\ B \leftarrow Y_1^2,\ C \leftarrow (X_1 + Y_1)^2,\ D \leftarrow A + B,$	$(3\mathbf{s})$
$E \leftarrow C - D,\ F \leftarrow B - A,\ G \leftarrow Z_1^2,\ H \leftarrow 2G - D,$	$(1\mathbf{s})$
$X_3 \leftarrow E \cdot H,\ Y_3 \leftarrow D \cdot F,\ Z_3 \leftarrow D \cdot H,\ I \leftarrow G \cdot F,\ J \leftarrow (I - Y_3) \cdot E$	$(5\mathbf{m})$
$K \leftarrow J \cdot (x/y - y/x),\ L \leftarrow (I - Y_3) \cdot 2F,\ M \leftarrow Z_3 \cdot dG$	$((2 + \frac{k}{2})\mathbf{m})$
$N \leftarrow M \cdot (xy)^2,\ P \leftarrow Z_3 \cdot (A + B - G),\ l_1 \leftarrow K + L + N - P$	$((1\mathbf{m} + \frac{k}{2})\mathbf{m})$
$f^{(4)} \leftarrow (f^{(4)})^2 l_1$	$(1\mathbf{M}{+}1\mathbf{S})$

case for Jacobian coordinates are detailed in Appendix A, while the complexity in the '$a = -3$' case is taken directly from [21], although some further $\mathbf{s} - \mathbf{m}$ tradeoffs might be possible.

Mixed addition Next, we take a look at the mixed addition step in a Miller iteration. We first count the number of operations that must be executed when

Table 3. Comparison of costs for the doubling step of the Miller operation in the case of k even

	$k = 2$	$k \geq 4$
Jacobian coordinates	$10\mathbf{s} + 3\mathbf{m} + \mathbf{S} + \mathbf{M}$	$11\mathbf{s} + (k+1)\mathbf{m} + \mathbf{S} + \mathbf{M}$
Jacobian coordinates for $a = -3$	$4\mathbf{s} + 8\mathbf{m} + \mathbf{S} + \mathbf{M}$	$4\mathbf{s} + (k+7)\mathbf{m} + \mathbf{S} + \mathbf{M}$
Das/Sarkar Edwards coordinates (supersingular curves)	$6\mathbf{s} + 9\mathbf{m} + \mathbf{S} + \mathbf{M}$	-
Das/Sarkar Edwards inverted coordinates (supersingular curves)	$6\mathbf{s} + 9\mathbf{m} + \mathbf{S} + \mathbf{M}$	-
Edwards coordinates	$4\mathbf{s} + 9\mathbf{m} + \mathbf{S} + \mathbf{M}$	$4\mathbf{s} + (k+8)\mathbf{m} + \mathbf{S} + \mathbf{M}$

adding $K = (X_1, Y_1, Z_1)$ and $P = (X_0, Y_0, 1)$. The result is $K + P = (X_3, Y_3, Z_3)$ with:

$$\begin{aligned} X_3 &= Z_1(X_0Y_1 + Y_0X_1)(Z_1^2 - dX_0X_1Y_0Y_1), \\ Y_3 &= Z_1(Y_0Y_1 - X_0X_1)(Z_1^2 + dX_0X_1Y_0Y_1), \\ Z_3 &= (Z_1^2 + dX_0X_1Y_0Y_1)(Z_1^2 - dX_0X_1Y_0Y_1). \end{aligned}$$

On the $E_{s,p}$, we consider $l_{s,p}$ the straight line passing through $\phi(K) = (p_1, s_1)$ and $\phi(P) = (p_0, s_0)$ and $v_{s,p}$ the vertical line passing through the point $\phi(K) + \phi(P) = (p_3, s_3)$. We get:

$$\begin{aligned} l_{s,p}(s,p) &= (p_0 - p_1)(s - s_1) - (s_0 - s_1)(p - p_1); \\ v_{s,p}(s,p) &= p - p_3. \end{aligned}$$

Replacing with the expressions of p_0, p_1, s_0, s_1 and multipling the equation above by (x_1y_1) we get:

$$\begin{aligned} l_{s,p}(s,p) = ((x_1y_1)^2 - (x_0y_0)^2)(x_1y_1(x/y - y/x) - (x_1^2 - y_1^2)) \\ -(x_1^2 - y_1^2 - x_1y_1(x_0^2 - y_0^2))((xy)^2 - (x_1y_1)^2). \end{aligned}$$

Consequently, we get normalized functions l and v of equations:

$$\begin{aligned} l(x,y) = l_1(x,y)/l_2 = ((X_1^2 + Y_1^2 - Z_1^2 - dZ_1^2(X_0Y_0)^2)(X_1Y_1(\frac{x}{y} - \frac{y}{x}) - \\ (X_1^2 - Y_1^2)) - (X_1^2 - Y_1^2 - X_1Y_1(\frac{X_0}{Y_0} - \frac{Y_0}{X_0})) \\ \cdot(dZ_1^2(xy)^2 - (X_1^2 + Y_1^2 - Z_1^2))) \\ /(X_1Y_1(X_1^2 + Y_1^2 - Z_1^2 - dZ_1^2(X_0Y_0)^2)); \\ v(x,y) = v_1(x,y)/v_2 = (dZ_3^2(xy)^2 - (X_3^2 + Y_3^2 - Z_3^2))/(X_3^2 + Y_3^2 - Z_3^2). \end{aligned}$$

For the same reasons as above, the mixed addition step in the case of even k becomes:

$$f^{(4)} \leftarrow (f^{(4)})^2 l_1(Q). \tag{11}$$

Detailed computations of the mixed addition step for $k > 2$ are presented in Table 4 (please check Appendix C for a proof of the correctness of these computations). We suppose having computed expressions depending only on P, the base point (i.e. X_0Y_0, $(X_0Y_0)^2$, $X_0/Y_0 - Y_0/X_0$) once and for all in the very beginning. The operation count gives $4\mathbf{s} + 15\mathbf{s} + \mathbf{M}$. As computing $X_0/Y_0 - Y_0/X_0$ costs one inversion in $\mathbb{F}_q$, in some cases it will be less expensive to work with $l_1^{'} = (X_0Y_0)l_1$ instead of l_1. For protocols in which Q is a fixed point, this will give an inversion free algorithm, but the mixed addition step will cost $4\mathbf{s} + 16\mathbf{m} + \mathbf{M}$ for $k = 2$ and $4\mathbf{s} + (k + 14)\mathbf{m}$ for $k \geq 4$.

Table 4. Operations of the mixed addition step of a Miller operation for $k > 2$

$A \leftarrow X_1^2, B \leftarrow Y_1^2, C \leftarrow (X_1 + Y_1)^2 - A - B, D \leftarrow C \cdot (dX_0Y_0)$	$(1\mathbf{m}+3\mathbf{s})$
$E \leftarrow 2(X_1 + X_0) \cdot (Y_0 + Y_1) - C - 2X_0Y_0,$	$(1\mathbf{m})$
$F \leftarrow 2(X_1 + Y_0) \cdot (Y_1 - X_0) - C + 2X_0Y_0, G \leftarrow Z_1^2$	$(1\mathbf{m}+1\mathbf{s})$
$X_3 \leftarrow Z_1 \cdot E \cdot (2G + D), Y_3 \leftarrow Z_1 \cdot F \cdot (2G - D), Z_3 \leftarrow (2G - D) \cdot (2G + D)$	$(5\mathbf{m})$
$H \leftarrow dG \cdot (X_0Y_0)^2, I \leftarrow (A + B - G - H) \cdot C,$	$(2\mathbf{m})$
$J \leftarrow I \cdot (x/y - y/x), K \leftarrow 2(A + B - G - H) \cdot (A - B)$	$((1 + \frac{k}{2})\mathbf{m})$
$L \leftarrow C \cdot (X_0/Y_0 - Y_0/X_0), M \leftarrow dG \cdot (2A - 2B - L), N \leftarrow M \cdot (xy)^2$	$((2 + \frac{k}{2})\mathbf{m})$
$P \leftarrow (2A - 2B - L) \cdot (A + B - G), l_1 \leftarrow J - K - N + P$	$(1\mathbf{m})$
$f^{(4)} \leftarrow (f^{(4)})^2 \cdot l_1$	$(1\mathbf{M})$

Results and performance comparison are summarized Table 5.

Table 5. Comparison of costs for the mixed addition step of the Miller operation in the case of k even

	$k = 2$	$k \geq 4$
Jacobian coordinates	$3\mathbf{s} + 11\mathbf{m} + \mathbf{M}$	$3\mathbf{s} + (k + 9)\mathbf{m} + 1\mathbf{M}$
Das/Sarkar Edwards coordinates (supersingular curves)	$1\mathbf{s} + 18\mathbf{m} + \mathbf{M}$	-
Das/Sarkar Edwards inverted coordinates (supersingular curves)	$1\mathbf{s} + 17\mathbf{m} + \mathbf{M}$	-
Edwards coordinates	$4\mathbf{s} + 15\mathbf{m} + \mathbf{M}$	$4\mathbf{s} + (k + 14)\mathbf{m} + 1\mathbf{M}$

Comparison By looking at Tables 3 and 5 one can see that in the case of an even embedding degree the cost of an implementation of Miller's algorithm in Edwards coordinates will be slightly more expensive than an implementation in Jacobian coordinates. We also checked performances of our method in inverted Edwards coordinates, but we did not obtain better results. We find it important to state that, no matter the representation one might choose to implement Miller's algorithm in high embedding degrees, it would be impossible to avoid the costly computation of $1\mathbf{M} + 1\mathbf{S}$ in equation (10) or the $1\mathbf{M}$ in equation (11),

as these are the updates in the Miller loop.

It is clear that when Edwards coordinates are prefered for the implementation of a protocol for certain reasons (scalar multiplication is faster, resistant to side channel attacks), a solution would be to switch to Jacobian coordinates and to compute the pairing on the Weierstrass form. Even though pairing implementation is faster in Jacobian coordinates, this approach will cost at least one field inversion. Consequently, on restricted devices, it is preferable to use our approach and avoid inversions.

5 Conclusion

In this paper, we have given a new algorithm to compute pairings on Edwards curves and compared its performance to that of an implementation of Miller's algorithm in Jacobian coordinates and to the method for Edwards coordinates from [13]. We showed that this algorithm is competitive and that if Edwards coordinates were to be chosen for the implementation of a pairing-based protocol, the presented approach gives an inversion-free algorithm, which is clearly un advantage on restricted devices.

6 Acknowledgements

The authors are grateful to Tanja Lange for helpful comments on earlier versions of this paper and to Vanessa Vitse for implementing pairings on Edwards curves in the first place.

References

1. Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted Edwards curves. In Serge Vaudenay, editor, *AFRICACRYPT*, volume 5023 of *Lecture Notes in Computer Science*, pages 389–405. Springer Verlag, 2008.
2. Daniel J. Bernstein and Tanja Lange. Explicit-formulas database, 2007. http://hyperelliptic.org/EFD/.
3. Daniel J. Bernstein and Tanja Lange. Faster addition and doubling on elliptic curves. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer Verlag, 2007.
4. Daniel J. Bernstein and Tanja Lange. Inverted Edwards coordinates. In Serdar Boztas and H.F. Lu, editors, *AAECC 2007*, volume 4851 of *Lecture Notes in Computer Science*, pages 20–27. Springer Verlag, 2007.
5. Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart. *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 2005.
6. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Verlag, 2001.

7. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer Verlag, 2001.
8. Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 04: 11th Conference on Computer and Communications Security*, pages 168–177. ACM Press, 2004.
9. Wieb Bosma. Signed bits and fast exponentiation. *J. de théorie des nombres de Bordeaux*, 13(1):27–41, 2001.
10. Friederike Brezing and Annegret Weng. Elliptic curves suitable for pairing based cryptography. *Des. Codes Cryptography*, 37(1):133–141, 2005.
11. Sanjit Chatterjee, Palash Sarkar, and Rana Barua. Efficient computation of Tate pairing in projective coordinate over general characteristic fields. *ICISC 2004*, 3506:168–181, 2005.
12. Zhaohui Cheng and Manos Nistazakis. Implementing pairing-based cryptosystems. *2nd International Workshop on Wireless Security Technologies IWWST-2004*, 2004.
13. M. Prem Laxman Das and Palash Sarkar. Pairing computation on twisted Edwards form elliptic curves. In Steven Galbraith and Kenny Paterson, editors, *Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 192–210. Springer Verlag, 2008.
14. Harold M. Edwards. A normal form for elliptic curves. *Bull. AMS*, 44:393–422, 2007.
15. David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive, Report 2006/372, 2006. http://eprint.iacr.org/.
16. Robert Granger, Florian Hess, Roger Oyono, Nicolas Thériault, and Frederik Vercauteren. Ate pairing on hyperelliptic curves. In Moni Naor, editor, *EUROCRYPT07*, volume 4515 of *Lecture Notes in Computer Science*, pages 419–436. Springer Verlag, 2007.
17. Robert Granger, Dan Page, and Nigel P. Smart. High security pairing-based cryptography revisited. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *ANTS VI*, volume 4076 of *Lecture Notes in Computer Science*, pages 480–494, 2006.
18. Robin Hartshorne. *Algebraic Geometry*, volume 52 of *Graduate texts in Mathematics*. Springer, 1977.
19. Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The Eta Pairing Revisited. *IEEE Transactions on Information Theory*, 52:4595–4602, 2006.
20. Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, September 2004.
21. Neal Koblitz and Alfred Menezes. Pairing-based cryptography at high security levels. In Nigel P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *Lecture Notes in Computer Science*, pages 13–36, 2005.
22. Victor S. Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235–261, September 2004.
23. Joseph H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate texts in Mathematics*. Springer, 1986.

A Pairing computation in Jacobian coordinates

One of the most efficient known ways of computing pairings on an elliptic curve given by a Weierstrass equation is to use Jacobian coordinates, as stated in [21] and in [17]. A point (X, Y, Z) in Jacobian coordinates represents the affine point $(X/Z^2, Y/Z^3)$ on the elliptic curve. In order to permit comparison with the results we obtained in section 4.1, we give formulae for the computation of the doubling step of Miller's algorithm, using more recent formulas for doubling on elliptic curves in Jacobian coordinates given in [2]. We suppose that the embedding degree k is even. We are interested in computing $T_r(P, Q)$, where P is chosen such that $\langle P \rangle$ is the unique subgroup of order r in $E(\mathbb{F}_q)$ and $Q \in E(\mathbb{F}_{q^k})$ is so that the x-coordinates of all its points lie in $\mathbb{F}_{q^{k/2}}$ and the y-coordinates are products of elements of $\mathbb{F}_{q^{k/2}}$ with some fixed squareroot $\sqrt{\beta}$ of a nonsquare element β in $\mathbb{F}_{q^{k/2}}$. We will look at the doubling step of the Miller operation for the i-th bit of r. In order to be consistent with notations in [12], we represent the point K as $K = (X_1, Y_1, Z_1, W_1)$, where (X_1, Y_1, Z_1) are the Jacobian coordinates of the point K on the Weierstrass curve and $W_1 = Z_1^2$. We compute $2K = (X_3, Y_3, Z_3, W_3)$ as

$$\begin{aligned} X_3 &= (3X_1^2 + aW_1^2)^2 - 8X_1Y_1^2, \\ Y_3 &= (3X_1^2 + aW_1^2)(4X_1Y_1^2 - X_3) - 8Y_1^4, \\ Z_3 &= 2Y_1Z_1 \\ W_3 &= Z_3^2. \end{aligned}$$

We write the normalized functions l and v that appear in (7) as $l = l_1/l_2$ and $v = v_1/v_2$:

$$\begin{aligned} l(x, y) &= l_1(x, y)/l_2 = (Z_3W_1y - 2Y_1^2 - (3X_1^2 + aW_1^2)(W_1x - X_1))/(Z_3W_1) \\ v(x, y) &= v_1(x, y)/v_2 = (W_3x - X_3)/W_3. \end{aligned}$$

As explained in [21], l_2, v_2 and $v_1(Q)$ can be ignored as they lie in proper subfields of $\mathbb{F}_{q^k}$. So in the doubling part of Miller's algorithm we perform the following operations:

$$\begin{aligned} K &\leftarrow 2K \\ f_1 &\leftarrow f_1^2 l_1(Q). \end{aligned}$$

For $k = 2$, $x \in \mathbb{F}_q$ so we can compute the function l_1 as:

$$l_1(x, y) = Z_3W_1y - 2Y_1^2 - (3X_1^2 + aW_1^2)(W_1x - X_1).$$

For $k > 2$ x is in $\mathbb{F}_{q^{k/2}}$, so the computation is slightly different:

$$l_1(x, y) = Z_3W_1y - 2Y_1^2 - W_1(3X_1^2 + aW_1^2)x + X_1(3X_1^2 + aW_1^2).$$

Computations detailed in Table 6 for $k > 2$ give an operation count of $11\mathbf{s} + (k + 1)\mathbf{m} + 1\mathbf{S} + 1\mathbf{M}$. The following lines give a Pari-Gp script which formally verifies that computations done in Table 6 are correct.

Table 6. Operations of the doubling part of a Miller operation for $k > 2$

$A \leftarrow W_1^2,\ B \leftarrow X_1^2,\ C \leftarrow Y_1^2,\ D \leftarrow C^2$	(4**s**)
$E \leftarrow (X_1 + C)^2 - B - D,\ F \leftarrow 3B + aA,\ G \leftarrow F^2$	(2**s**)
$X_3 \leftarrow -4E + G,\ Y_3 \leftarrow -8D + F \cdot (2E - X_3),\ Z_3 \leftarrow (Y_1 + Z_1)^2 - C - W_1$	(1**m**+1**s**)
$W_3 \leftarrow Z_3^2,\ H \leftarrow (Z_3 + W_1)^2 - W_3 - A,\ I \leftarrow H \cdot y$	($\frac{k}{2}$**m**+2**s**)
$J \leftarrow (F + W_1)^2 - G - A,\ K \leftarrow J \cdot x,\ L \leftarrow (F + X_1)^2 - G - B$	($\frac{k}{2}$**m**+2**s**)
$l_1 \leftarrow I - 4C - K + L$	
$f_1 \leftarrow f_1^2 \cdot l_1$	(1**M**+1**S**)

```
W1=Z1^2;A=W1^2;B1=X1^2;C=Y1^2;
D=C^2;E=(X1+C)^2-B1-D;
F=3*B1+a*A;G=F^2;X3=-4*E+G;
Y3=-8*D+F*(2*E-X_3);Z_3=(Y_1+Z_1)^2-C-Z1^2;
W3=Z3^2;H=(Z3+W1)^2-W3-A;
I1=H*y;J=(F+W1)^2-G-A;
K=J*x;L=(F+X1)^2-G-B1;
l1=I1-4*C-K+L
l1verif=Z3*(Z1^2)*y-2*Y1^2-(Z1^2)*(3*(X1^2)+a*(Z1^4))*x+X1*(3*(X1^2)+a*Z1^4)
l1-2*l1verif
```

B Pari-Gp script for the doubling part of Miller's algorithm in Edwards coordinates

The following script checks computations in Table 2:

```
A=X1^2;B=Y1^2;C=(X1+Y1)^2;
D=A+B;E=C-D;F=B-A;
G=Z1^2;H=2*G-D;X3=E*H;
Y3=D*F;Z3=D*H;I1=G*F;
J=(I1-Y3)*E;K=J*(x/y-y/x);
L=(I1-Y3)*2*F;M=Z3*d*G;
N=M*(x*y)^2;P=Z3*(A+B-G);
l1=K+L+N-P;
l1
l1verif=((X1^2+Y1^2-Z1^2)*(X1^2-Y1^2))*2*X1*Y1*(x/y-y/x);
l1verif=l1verif-((X1^2+Y1^2-Z1^2)*(X1^2-Y1^2))*(2*(X1^2-Y1^2));
l1verif=l1verif+Z3*(d*(Z1^2)*((x*y)^2))-Z3*(X1^2+Y1^2-Z1^2)
l1-l1verif
```

C Pari-Gp script for the mixed addition part of Miller's algorithm

The following script checks the computations in Table 4:

```
A=X1^2;B=Y1^2;C=(X1+Y1)^2-A-B;
D=C*(d*X0*Y0);E=2*(X1+Y1)^2-A-B;
D=C*d*X0*Y0;E=2*(X1+X0)*(Y0+Y1)-C-2*X0*Y0;
F=2*(X1+Y0)*(Y1-X0)-C-2*X0*Y0;
G=Z1^2;X3=Z1*E*(2*G+D);
Y3=Z1*F*(2*G-D);Z3=(2*G-D)*(2*G+D);
H=d*G*(X0*Y0)^2;I1=(A+B-G-H)*C;
J=I1*(x/y-y/x);K=2*(A+B-G-H)*(A-B);
L=C*(X0/Y0-Y0/X0);M=d*G*(2*A-2*B-L);
N=M*(x*y)^2;P=(2*A-2*B-L)*(A+B-G);
l1verif=(X1^2+Y1^2-Z1^2-d*(Z1^2)*(X0*Y0)^2)*(X1*Y1*(x/y-y/x)-(X1^2-Y1^2));
l1verif=l1verif-(X1^2-Y1^2-X1*Y1*(X0/Y0-Y0/X0))*(d*(Z1^2)*((x*y)^2)-(X1^2+Y1^2-Z1^2))
l1=J-K-N+P
l1-2*l1verif
```