AAECC (2016) 27:205–217
DOI 10.1007/s00200-015-0278-z

ORIGINAL PAPER

Further refinements of Miller's algorithm on Edwards curves

Duc-Phong Le[1] · **Chik How Tan**[1]

Received: 30 December 2014 / Revised: 23 October 2015 / Accepted: 26 October 2015 /
Published online: 4 November 2015

Abstract Recently, Edwards curves have received a lot of attention in the cryptographic community due to their fast scalar multiplication algorithms. Then, many works on the application of these curves to pairing-based cryptography have been introduced. In this paper, we investigate refinements to Miller's algorithm that play a central role in paring computation. We first introduce a variant of Miller function that leads to a more efficient variant of Miller's algorithm on Edwards curves. Then, based on the new Miller function, we present a refinement to Miller's algorithm that significantly improves the performance in comparison with the original Miller's algorithm. Our analyses also show that the proposed refinement is approximately 25 % faster than Xu–Lin's refinements (CT-RSA, 2010). Last but not least, our approach is *generic*, hence the proposed algorithms allow to compute *both* Weil and Tate pairings on pairing-friendly Edwards curves of *any* embedding degree.

Keywords Miller's algorithm · Pairing computation · Edwards curves · Tate/Weil pairings

Mathematics Subject Classification 11T71

✉ Duc-Phong Le
tslld@nus.edu.sg

Chik How Tan
tsltch@nus.edu.sg

[1] Temasek Laboratories, National University of Singapore, 5A Engineering Drive 1, #09-02, Singapore 117411, Singapore

1 Introduction

In 2007, Bernstein and Lange [7] introduced Edwards curves to cryptography. Their study and subsequent works in [2,8,9,16] showed that the addition law on such curves is more efficient than all previously known formulas. Edwards curves have thus attracted great interest in applications that require elliptic curve operations to achieve faster arithmetic. Subsequently, the application of Edwards curves to pairing-based cryptography has been studied in several research papers [1,13,17,20,24]. Although, pairing computation on Edwards curves is slightly slower than on Weierstrass curves so far. However, in many pairing-based cryptosystems, the most time-consuming operation is still to compute scalar multiples aP of a point P, and it thus may be advantageous to use Edwards curves in these applications.

Pairing (or *bilinear map*) is probably the most useful cryptographic tool in the 2000s. It was first introduced to cryptography in Joux's seminal paper [18] in 2000 that describes a tripartite (bilinear) Diffie–Hellman key exchange. Then, the use of cryptosystems based on pairings has had a huge success with some notable breakthroughs such as the first practical identity-based encryption scheme [4].

Miller's algorithm [21,22] plays an important role in pairing computations on elliptic curves. Many papers are devoted to improvements in its efficiency. For example, using specific pairing-friendly elliptic curves to speed up Miller's algorithm [5,10,12]. Another approach of improving Miller's algorithm is to reduce the Miller-loop length by introducing variants of Tate pairings, for example Eta pairing [6], Ate pairing [15], and particular *optimal pairings* [14,23]. For a more generic approach, studies in [3,11,19] improved the performance for computing pairings of any type (i.e., Weil, Tate, optimal pairings), and on *generic* pairing-friendly elliptic curves.

Basically, Miller's algorithm is based on a rational function g of three points P_1, P_2, P_3. This function is called Miller function and has its divisor $div(g) = (P_1) + (P_2) - (P_3) - (\mathcal{O})$, where $\mathcal{O}$ is a distinguished point (i.e. the neutral element of the group law). For curves of Weierstrass form, this function is defined to be the line passing through points P_1 and P_2 divided by the vertical line passing through the point P_3, where $P_3 = [P_1 + P_2]$. On Edwards curves, finding such a point P_3 is not straightforward as in Weierstrass curves because Edwards equation has degree 4, i.e. any line has 4 intersections with the curves instead of 3 on Weierstrass curves.

In [1], Arene et al. presented the first geometric interpretation of the group law on Edwards curves and showed how to compute Tate pairing on twisted Edwards curves by using a conic of degree 2. They also introduced explicit formulas with a focus on curves having an even *embedding degree*[1]. Then, Xu and Lin [24] proposed refinements to Miller's algorithm that sped up the pairing computation on *generic* Edwards curves, i.e. curves with arbitrary embedding degree. The cost of their refinements is about 76.8 % of that of the original Miller's algorithm on Edwards curves.

In this paper, we further extend Xu–Lin's works and propose new refinements to Miller's algorithm. Similar to Xu–Lin's refinements, our approach is *generic*. Although

[1] Let E be an elliptic curve defined over a prime finite field $\mathbb{F}_p$, and r be a prime dividing $\#E(\mathbb{F}_p)$. The embedding degree of E with respect to r is the smallest positive integer k such that $r|p^k - 1$. In other words, k is the smallest integer such that $\mathbb{F}_{p^k}^*$ contains r-roots of unity.

this approach did not bring a dramatic efficiency as that of Arene et al. [1] when computing Tate pairing over Edwards curves with *even* embedded degrees, the proposed refinements can be used to compute pairings of *any* type over pairing-friendly Edwards curves with *any* embedding degree. This approach is of particular interest to compute optimal pairings [14,23], and in situations where the denominator elimination technique using a twist is not possible (e.g., Edwards curves with *odd* embedding degrees).[2] We also analyze and show that our new algorithm is faster than the original Miller algorithm and Xu–Lin's refinements. When the embedding degree k gets larger, the time required to perform operations in the full extension field dominates that in the base field, then our algorithm is faster by approximately 25 % than Xu–Lin's algorithm.

The paper is organized as follows. In Sect. 2, we briefly recall some background on pairing computation on Edwards curves, and then Xu–Lin's refinements. In Sect. 3, we present our refinements of Miller's algorithm. Section 4 gives some discussion on performance of the proposed algorithms. Section 5 is our conclusion.

2 Preliminaries

2.1 Pairings on Edwards curves

Let $\mathbb{F}_p$ be a prime finite field, where p is a prime different from 2. A *twisted Edwards curve* $E_{a,d}$ defined over $\mathbb{F}_p$ is the set of solutions (x, y) of the following *affine equation*:

$$E_{a,d} : ax^2 + y^2 = 1 + dx^2y^2, \tag{1}$$

where $a, d \in \mathbb{F}_p^*$, and $a \neq d$. Let $P_1 = (x_1, y_1)$ be a point on $E_{a,d}$. The negative of P is $-P = (-x_1, y_1)$. The point $\mathcal{O} = (0, 1)$ is the neutral element of the addition law. The point $\mathcal{O}' = (0, -1)$ has order 2. The points at infinity $\Omega_1 = (1 : 0 : 0)$ and $\Omega_2 = (0 : 1 : 0)$ are singular. The studies in [2,9,16] showed that twisted Edwards curves provide the fastest doubling and addition operations in elliptic curve cryptography.

The cryptographic pairing is a bilinear map that takes two points on elliptic curves defined over finite fields and returns a value in extension finite field. Let r be a prime number different from p and $r|\#E_{a,d}(\mathbb{F}_p)$, where $\#E_{a,d}(\mathbb{F}_p)$ is the number of points on the Edwards curve $E_{a,d}$. Let k be the embedding degree of the elliptic curve $E_{a,d}$ with respect to r, and let $E_{a,d}[r]$ denote the subgroup of points of order r on the Edwards curves $E_{a,d}$. A cryptographic pairing is defined over an Edwards curves $E_{a,d}$ as:

$$e : E_{a,d}[r] \times E_{a,d}[r] \to \mathbb{F}_{p^k}^*$$

[2] Note that by definition optimal pairings only require about $\log_2(r)/\varphi(k)$ iterations of the basic loop, where r is the group order, φ is Euler's totient function, and k is the embedding degree. For example, when k is prime, then $\varphi(k) = k - 1$. If we choose a curve having embedding degree $k \pm 1$, then $\varphi(k \pm 1) \leq \frac{k+1}{2}$ which is roughly $\frac{\varphi(k)}{2} = \frac{k-1}{2}$, so that at least twice as many iterations are necessary if curves with embedding degrees $k \pm 1$ are used instead of curves of embedding degree k.

The key to the definition of pairings is the evaluation of rational functions in divisors. In this section, we briefly recall Miller's algorithm [22] that is so far the best known method to compute pairings over (hyper-)elliptic curves. Readers who want to know more about pairings can refer to papers [22,23]. Let $P, Q \in E_{a,d}$ be two points of order r. The main part of Miller's algorithm is to construct the rational function $f_{r,P}$ and evaluating $f_{r,P}(Q)$ with $div(f_{r,P}) = r(P) - (rP) - [r-1](\mathcal{O})$.

Let m and n be two integers, and $g_{mP,nP}$ be a rational function, so-called *Miller function*, whose divisor $div(g_{mP,nP}) = (mP) + (nP) - ([m+n]P) - (\mathcal{O})$. Miller's algorithm is based on the following lemma.

Lemma 1 *(Lemma 2, [22]) For n and m two integers, up to a multiplicative constant, we have*

$$f_{m+n,P} = f_{m,P} f_{n,P} g_{mP,nP}. \tag{2}$$

Equation (2) is called *Miller relation*, which is proved by considering divisors. Miller's algorithm makes use of Lemma 1 with $m = n$ in a doubling step and $n = 1$ in an addition step. For Edwards curves, Arene et al. [1] defined the Miller function in the following theorem.

Theorem 1 *(Theorem 2, [1]) Let $a, d \in \mathbb{F}_p^*$, $a \neq d$ and $E_{a,d}$ be a twisted Edwards curve over $\mathbb{F}_p$. Let P_1, $P_2 \in E_{a,d}(\mathbb{F}_p)$. Define $P_3 = P_1 + P_2$. Let ϕ_{P_1,P_2} be the equation of the conic $\mathscr{C}$ passing through $P_1, P_2, -P_3, \Omega_1, \Omega_2, \mathcal{O}'$ whose divisor is $(P_1) + (P_2) + (-P_3) + (\mathcal{O}') - 2(\Omega_1) - 2(\Omega_2)$. Let ℓ_{1,P_3} be the horizontal line going through P_3 whose divisor is $div(\ell_{1,P_3}) = (P_3) + (-P_3) - 2(\Omega_2)$, and $\ell_{2,\mathcal{O}}$ is the vertical line going through $\mathcal{O}$ whose divisor is $(\mathcal{O}) + (\mathcal{O}') - 2(\Omega_1)$. Then we have*

$$div\left(\frac{\phi_{P_1,P_2}}{\ell_{1,P_3}\ell_{2,\mathcal{O}}}\right) \sim (P_1) + (P_2) - (P_3) - (\mathcal{O}). \tag{3}$$

The rational function $g_{P_1,P_2} = \frac{\phi_{P_1,P_2}}{\ell_{1,P_3}\ell_{2,\mathcal{O}}}$ consists of three terms and can be thus considered as Miller function on Edwards curves. Miller's algorithm for Edwards curves using this function works as in Algorithm 1.

2.2 Xu–Lin's refinements

For simplicity, we make use of the notation $\phi_{P,P}$ (resp. $\ell_{2,[2]P}$, and $\ell_{1,\mathcal{O}}$) replacing for $\phi_{P,P}(Q)$ (resp. $\ell_{2,[2]P}(Q)$, and $\ell_{1,\mathcal{O}}(Q)$). By extending the Blake et al.'s method [11] to Edwards curves, Xu and Lin [24] presented a refinement of Miller's algorithm. Their algorithm was derived from the following theorem.

Theorem 2 (Theorem 1, *[24]) Let $E_{a,d}$ be a twisted Edwards curve over $\mathbb{F}_p$ and P, $R \in E_{a,d}$ be two points of order r. Then,*

1.

$$\left(\frac{\phi_{R,R}}{\ell_{1,[2]R}\ell_{2,\mathcal{O}}}\right)^2 \frac{\phi_{[2]R,[2]R}}{\ell_{1,[4]R}\ell_{2,\mathcal{O}}} = \frac{\phi_{R,R}^2}{\phi_{[-2]R,[-2]R}\phi_{\mathcal{O},\mathcal{O}}};$$

Input: $r = \sum_{i=0}^{t} r_i 2^i$ with $r_i \in \{0, 1\}$, $P, Q \in E[r]$;
Output: $f = f_{r,P}(Q)$;

$R \leftarrow P$, $f \leftarrow 1$, $g \leftarrow 1$
for $i = t - 1$ **to** 0 **do**
  $f \leftarrow f^2 \cdot \phi_{R,R}(Q)$
  $g \leftarrow g^2 \cdot \ell_{1,2R}(Q) \cdot \ell_{2,\mathcal{O}}(Q)$
  $R \leftarrow 2R$
  if $(r_i = 1)$ **then**
    $f \leftarrow f \cdot \phi_{R,P}(Q)$
    $g \leftarrow g \cdot \ell_{1,R+P}(Q) \cdot \ell_{2,\mathcal{O}}(Q)$
    $R \leftarrow R + P$
  end
end
return f/g

Algorithm 1: Miller's Algorithm for twisted Edwards curves [24]

2.

$$\frac{\phi_{R,R}}{\ell_{1,[2]R}\ell_{2,\mathcal{O}}}\frac{\phi_{[2]R,P}}{\ell_{1,[2]R+P}\ell_{2,\mathcal{O}}} = \frac{\phi_{R,R}\ell_{1,P}}{\phi_{[2]R+P,-P}\ell_{2,\mathcal{O}}}.$$

The above theorem was proven by calculating divisors (see [24] for more details). From this theorem, they described a variant of Miller's algorithm in radix-4 [24, Algorithm 3]. They also claimed that the total cost of the proposed algorithm is about 76.8 % of that of the original Miller algorithm. In the following section, we will present our refinements that are approximately 25 % faster than Xu–Lin's refinements.

3 Our improvements on Miller's algorithm

3.1 First improvement

We first present a new rational function whose divisor is equivalent to Miller function (Eq. (3)) presented in [1].

Definition 1 Let $E_{a,d}$ be a twisted Edwards curve and $R, \ P \in E_{a,d}$. Let $\phi_{R,P}$ be a conic passing through R and P, and let $\phi_{R+P,-[R+P]}$ a conic passing through $R + P$ and $-[R + P]$. Then we define

$$g_{R,P} = \frac{\phi_{R,P}}{\phi_{R+P,-[R+P]}}. \tag{4}$$

Lemma 2 *We have*

$$div(g_{R,P}) = (R) + (P) - ([R + P]) - \mathcal{O}.$$

Proof By calculating divisors, it is straightforward to see that

$$\text{div}(g_{R,P}) = (R) + (P) + (-[R + P]) + (\mathcal{O}') - 2(\Omega_1) - 2(\Omega_2)$$

$$
\begin{aligned}
&-([R+P])-(-[R+P])-(\mathcal{O})-(\mathcal{O}')+2(\Omega_1)+2(\Omega_2)\\
&=(R)+(P)-([R+P])-(\mathcal{O})\ .
\end{aligned}
$$

which concludes the proof. □

Recall that the Miller function is defined as $g_{R,P}=\frac{\ell_{R,P}}{\upsilon_{R+P}}$ on Weierstrass curves, where $\ell_{R,P}$, υ_{R+P} are lines passing through R, P and $R+P$, $-[R+P]$, respectively (see [22]). Eq. (4) thus looks like the Miller function on Weierstrass curves if a conic plays role of a line function. Notice that during pairing computation, R is multiple of the inputted point P. By applying Eq. (4), a variant of Miller's algorithm is described in Algorithm 2.

Input: $r=\sum_{i=0}^{t} r_i 2^i$ with $r_i \in \{0,1\}$, $P, Q \in E[r]$;
Output: $f = f_r(Q)$;
$R \leftarrow P$, $f \leftarrow 1$, $g \leftarrow 1$
for $i = t-1$ **to** 0 **do**
1 $\quad f \leftarrow f^2 \cdot \phi_{R,R}(Q)$
2 $\quad g \leftarrow g^2 \cdot \phi_{2R,-2R}(Q)$
$\quad R \leftarrow 2R$
$\quad$ **if** $(r_i = 1)$ **then**
3 $\quad\quad f \leftarrow f \cdot \phi_{R,P}(Q)$
4 $\quad\quad g \leftarrow g \cdot \phi_{R+P,-(R+P)}(Q)$
$\quad\quad R \leftarrow R+P$
$\quad$ **end**
end
return f/g

Algorithm 2: First improvement of Miller's Algorithm for twisted Edwards curves

Remark 1 As the original Miller algorithm, our algorithm cannot avoid divisions needed to update f. But we can reduce them easily to one inversion at the end of the addition chain (for the cost of one squaring, and one extra multiplication if $r_i = 1$ per iteration of the algorithm).

Algorithm 2 always performs one doubling step per iteration and one addition step if $r_i = 1$. To update functions f, g, the doubling step requires 2 squarings and 2 multiplications in the full extension field $\mathbb{F}_{p^k}$ and the addition step requires 2 multiplications. It is obvious to see that Algorithm 2 requires only one multiplication for updating the function g in doubling/addition steps instead of two multiplications as in Algorithm 1. Notice that this multiplication is costly because it is performed in the full extension field. In Sect. 4, we will provide a detailed analysis on the performance of these algorithms. In the following section, we introduce a further refinement that even offers a better performance in comparison with Algorithm 2.

3.2 Further refinement

The new refinement is inspired by the following lemmas.

Lemma 3 *We have*

$$g_{R,P} = \frac{\phi_{R,-R} \cdot \phi_{P,-P}}{\phi_{-R,-P} \cdot \phi_{\mathcal{O},\mathcal{O}}}. \tag{5}$$

Proof Again, this lemma is proved by considering divisors. Indeed,

$$\begin{aligned}
\operatorname{div}\left(\frac{\phi_{R,-R}\cdot\phi_{P,-P}}{\phi_{-R,-P}\cdot\phi_{\mathcal{O},\mathcal{O}}}\right) &= (R) + (-R) + (\mathcal{O}) + (\mathcal{O}') - 2(\Omega_1) - 2(\Omega_2) \\
&\quad + (P) + (-P) + (\mathcal{O}) + (\mathcal{O}') - 2(\Omega_1) - 2(\Omega_2) \\
&\quad - (-R) - (-P) - ([R+P]) - (\mathcal{O}') + 2(\Omega_1) + 2(\Omega_2) \\
&\quad - 3(\mathcal{O}) - (\mathcal{O}') + 2(\Omega_1) + 2(\Omega_2) \\
&= (R) + (P) - ([R+P]) - (\mathcal{O}) \\
&= \operatorname{div}(g_{R,P}) .
\end{aligned}$$

which concludes the proof. □

Lemma 4 *Let $R \in E_{a,d}$ be a point of order r, we have*

$$\frac{\phi_{R,R}}{\phi_{R,-R}^2 \cdot \phi_{2R,-2R}} = \frac{1}{\phi_{-R,-R} \cdot \phi_{\mathcal{O},\mathcal{O}}}. \tag{6}$$

Proof This lemma is easy to be proven using Definition 1 and Lemma 3 by replacing P by R. By Definition 1, we have

$$g_{R,R} = \frac{\phi_{R,R}}{\phi_{2R,-2R}},$$

and by Lemma 3, we have

$$g_{R,R} = \frac{\phi_{R,-R} \cdot \phi_{R,-R}}{\phi_{-R,-R} \cdot \phi_{\mathcal{O},\mathcal{O}}}.$$

Thus, we obtain

$$\frac{\phi_{R,R}}{\phi_{R,-R}^2 \cdot \phi_{2R,-2R}} = \frac{1}{\phi_{-R,-R} \cdot \phi_{\mathcal{O},\mathcal{O}}}$$

□

Remark 2 The right-hand side of Eq. (6) can be further simplified. Since Q is a fixed point during pairing computation, the conic function $\phi_{\mathcal{O},\mathcal{O}}$ can be precomputed and integrated into $\phi_{-R,-R}$ as follows:

$$\begin{aligned}
\phi'_{-R,-R}(Q) &= \phi_{-R,-R}(Q) \cdot \phi_{\mathcal{O},\mathcal{O}}(Q) \\
&= (c_{Z^2}(Z_Q^2 + Y_Q Z_Q) + c_{XY} X_Q Y_Q + c_{XZ} X_Q Z_Q)(X_Q(Z_Q - Y_Q)) \\
&= c_{Z^2}\eta_1 + c_{XY}\eta_2 + c_{XZ}\eta_3,
\end{aligned}$$

where $\eta_1 = (Z_Q^2 + Y_Q Z_Q)(X_Q(Z_Q - Y_Q))$, $\eta_2 = X_Q^2 Y_Q(Z_Q - Y_Q)$, $\eta_3 = X_Q^2 Z_Q(Z_Q - Y_Q)$. Because Q is fixed for the whole computation, the values η_1, η_2, and η_3 are also fixed. Hence, these values can be precomputed and stored. Eq. (6) can thus be rewritten as follows:

$$\frac{\phi_{R,R}}{\phi_{R,-R}^2 \cdot \phi_{2R,-2R}} = \frac{1}{\phi'_{-R,-R}}. \tag{7}$$

The right-hand side of Eq. (7) has only one conic function while the left-hand side contains 3 conic functions. We aim to use this observation to reduce $\phi_{R,-R}$ in Algorithm 2, where R is multiple of P. To do so, we apply the following simple strategy. Whenever possible, we delay $\phi_{R,-R}$ in the denominator for the next iteration. This conic function will be squared in the next doubling step, then by applying Eq. (7) one can cancel out $\phi_{R,-R}^2$ and $\phi_{2R,-2R}$.

We introduce a variable m to determine whether there exists a delay of one conic function $\phi_{R,-R}$ in the current iteration. If $m = 1$, there is a delay from the previous iteration, otherwise, there is no delay. Let $(r_t r_{t-1} \dots r_0)_2$, where $r_i \in \{0, 1\}$ be the binary representation of the order r. There are four cases to be considered as follows:

1. If the current bit $r_i = 0$ and $m = 0$. The proposed algorithm will perform a doubling step by applying Eq. (4), but keep delaying $\phi_{2R,-2R}$ for the next iteration, i.e. m will be set to 1 after this iteration.
2. If the current bit $r_i = 0$ and $m = 1$. The proposed algorithm will perform a doubling step by applying Eq. (7). This allows to eliminate the delayed conic $\phi_{R,-R}^2$ and $\phi_{2R,-2R}$. The value of m will be set to 0. This is only case in the proposed algorithm there is no delay for the next iteration.
3. If the current bit $r_i = 1$ and $m = 1$. The proposed algorithm will apply Eq. (7) in the doubling step and Eq. (4) in the addition step. It also keeps delaying $\phi_{2R+P,-(2R+P)}$ for the next iteration.
4. If the current bit $r_i = 1$ and $m = 0$. This is the most costly case of the proposed algorithm. The proposed algorithm will apply Eq. (4) in both doubling and addition steps. Likewise, it keeps delaying $\phi_{2R+P,-(2R+P)}$ for the next iteration.

Let $f_{r,P}^{(i)}$ be the value of the rational function $f_{r,P}$ when processing at the bit r_i of the order r. Formally, we define the following function $f_{r,P}$ by applying Eq. (7) as follows:

$$f_{r,P}^{(i)} = \begin{cases} (f_{r,P}^{(i+1)})^2 \phi_{R,R} & \text{if } r_i = 0 \text{ and } m = 0 \\ (f_{r,P}^{(i+1)})^2 \frac{1}{\phi'_{-R,-R}} & \text{if } r_i = 0 \text{ and } m = 1 \\ (f_{r,P}^{(i+1)})^2 \frac{\phi_{2R,P}}{\phi'_{-R,-R}} & \text{if } r_i = 1 \text{ and } m = 1 \\ (f_{r,P}^{(i+1)})^2 \frac{\phi_{R,R} \cdot \phi_{2R,P}}{\phi_{2R,-2R}} & \text{if } r_i = 1 \text{ and } m = 0, \end{cases}$$

where $0 \le i < t$, $f_{r,P}^{(t)} = 1$, and R is a multiple of the input point P at the current iteration, i.e. $R = [(r_t \dots r_{i+1})_2]P$. The proposed algorithm is described by the pseudo-code in Algorithm 3.

Input: $r = \sum_{i=0}^{t} r_i 2^i$ with $r_i \in \{0,1\}$, $P, Q \in E[r]$;
Output: $f = f_{r,P}(Q)$;

$T \leftarrow P$, $f \leftarrow 1$, $g \leftarrow 1$, $m \leftarrow 0$;
for $i = t-1$ **to** 0 **do**
1 **if** $(r_i = 0) \wedge (m = 0)$ **then**
$f \leftarrow f^2 \cdot \phi_{R,R}$; $g \leftarrow g^2$; $R \leftarrow 2R$; $m \leftarrow 1$
end
2 **if** $(r_i = 0) \wedge (m = 1)$ **then**
$f \leftarrow f^2$; $g \leftarrow g^2 \cdot \phi'_{-R,-R}$; $R \leftarrow 2R$; $m \leftarrow 0$
end
3 **if** $(r_i = 1) \wedge (m = 1)$ **then**
$f \leftarrow f^2 \cdot \phi_{2R,P}$; $g \leftarrow g^2 \cdot \phi'_{-R,-R}$; $R \leftarrow 2R + P$; $m \leftarrow 1$
end
4 **if** $(r_i = 1) \wedge (m = 0)$ **then**
$f \leftarrow f^2 \cdot \phi_{R,R} \cdot \phi_{2R,P}$; $g \leftarrow g^2 \cdot \phi_{2R,-2R}$; $R \leftarrow 2R + P$; $m \leftarrow 1$
end
end
if $m = 1$ **then**
$g \leftarrow g \cdot \phi_{R,-R}$
end
return $\frac{f}{g}$

Algorithm 3: Improved Refinement of Miller's Algorithm for any Pairing-Friendly Edwards Curve

3.3 Examples

We give some examples in this section. We list the calculation formulas of $f_{r,P}$ for $r = 5$, 17 and 23 using Miller's algorithm from [1,24] (Algorithm 1), our improved version 1 (Algorithm 2) and our improved version 2 (Algorithm 3). Here the symbol $\phi_{mP,nP}$, $\phi'_{mP,nP}$, $\ell_{1,nP}$, $\ell_{2,nP}$ and $f_{r,P}$ are shortened to $\phi_{m,n}$, $\phi'_{m,n}$, $\ell_{1,n}$, $\ell_{2,n}$ and f_r respectively, where $m, n \in \mathbb{Z}$.

Compute f_5:

Number	$(101)_2 = 5$
Algorithm 1	$f_5 = \frac{\phi_{1,1}^2}{\ell_{1,2}^2 \cdot \ell_{2,\mathcal{O}}^2} \frac{\phi_{2,2}}{\ell_{1,4} \cdot \ell_{2,\mathcal{O}}} \frac{\phi_{4,1}}{\ell_{1,5} \cdot \ell_{2,\mathcal{O}}}$
Algorithm 2	$f_5 = \frac{\phi_{1,1}^2}{\phi_{2,-2}^2} \frac{\phi_{2,2}}{\phi_{4,-4}} \frac{\phi_{4,1}}{\phi_{5,-5}}$
Algorithm 3	$f_5 = \phi_{1,1}^2 \frac{\phi_{4,1}}{\phi'_{-2,-2}} \frac{1}{\phi_{5,-5}}$

Compute f_{17}:

Number	$(10001)_2 = 17$
Algorithm 1	$f_{17} = \frac{\phi_{1,1}^8}{\ell_{1,2}^8 \cdot \ell_{2,\mathcal{O}}^8} \frac{\phi_{2,2}^4}{\ell_{1,4}^4 \cdot \ell_{2,\mathcal{O}}^4} \frac{\phi_{4,4}^2}{\ell_{1,8}^2 \cdot \ell_{2,\mathcal{O}}^2} \frac{\phi_{8,8}}{\ell_{1,16} \cdot \ell_{2,\mathcal{O}}} \frac{\phi_{16,1}}{\ell_{1,17} \cdot \ell_{2,\mathcal{O}}}$
Algorithm 2	$f_{17} = \frac{\phi_{1,1}^8}{\phi_{2,-2}^8} \frac{\phi_{2,2}^4}{\phi_{4,-4}^4} \frac{\phi_{4,4}^2}{\phi_{8,-8}^2} \frac{\phi_{8,8}}{\phi_{16,-16}} \frac{\phi_{16,1}}{\phi_{17,-17}}$
Algorithm 3	$f_{17} = \phi_{1,1}^8 \frac{1}{(\phi'_{-2,-2})^4} \phi_{4,4}^2 \frac{\phi_{16,1}}{\phi'_{8,-8}} \frac{1}{\phi_{17,-17}}$

Compute f_{23}:

Number	$(10111)_2 = 23$
Algorithm 1	$f_{23} = \frac{\phi_{1,1}^8}{\ell_{1,2}^8 \cdot \ell_{2,\mathcal{O}}^8} \frac{\phi_{2,2}^4}{\ell_{1,4}^4 \cdot \ell_{2,\mathcal{O}}^4} \frac{\phi_{4,1}^4}{\ell_{1,5}^4 \cdot \ell_{2,\mathcal{O}}^4} \frac{\phi_{5,5}^2}{\ell_{1,10}^2 \cdot \ell_{2,\mathcal{O}}^2} \frac{\phi_{10,1}^2}{\ell_{1,11}^2 \cdot \ell_{2,\mathcal{O}}^2} \frac{\phi_{11,11}}{\ell_{1,22} \cdot \ell_{2,\mathcal{O}}} \frac{\phi_{22,1}}{\ell_{1,23} \cdot \ell_{2,\mathcal{O}}}$
Algorithm 2	$f_{23} = \frac{\phi_{1,1}^8}{\phi_{2,-2}^8} \frac{\phi_{2,2}^4}{\phi_{4,-4}^4} \frac{\phi_{4,1}^4}{\phi_{5,-5}^4} \frac{\phi_{5,5}^2}{\phi_{10,-10}^2} \frac{\phi_{10,1}^2}{\phi_{11,-11}^2} \frac{\phi_{11,11}}{\phi_{22,-22}} \frac{\phi_{22,1}}{\phi_{23,-23}}$
Algorithm 3	$f_{23} = \phi_{1,1}^8 \frac{\phi_{4,1}^4}{(\phi'_{-2,-2})^4} \frac{\phi_{10,1}^2}{(\phi'_{-5,-5})^2} \frac{\phi_{22,1}}{\phi'_{-11,-11}} \frac{1}{\phi_{23,-23}}$

4 Performance discussion

In this section, we make a comparison between our refinements and the algorithms described in [1,24]. Before analyzing the costs of algorithm, we introduce notations for the cost of field arithmetic operations. Let $\mathbb{F}_{p^k}$ be an extension of degree k of $\mathbb{F}_p$ for $k \geq 1$ and let $\mathbf{I}_{p^k}$, $\mathbf{M}_{p^k}$, and $\mathbf{S}_{p^k}$ be the costs for inversion, multiplication, and squaring in the field $\mathbb{F}_{p^k}$, respectively.

The cost of the algorithms for pairing computation consists of three parts: the cost of updating the functions f, and g; the cost of updating the point R; and the cost of evaluating rational functions at some point Q. Arene et al. in [1, Section 5] analyzed in detail the total cost of updating the point R and coefficients c_{Z^2}, c_{XY}, and c_{ZZ} of the conic. Without special treatment, this cost is the same for all algorithms.

The most costly operations in pairing computations are operations in the full extension field $\mathbb{F}_{p^k}$. At high levels of security (i.e. k large), the complexity of operations in $\mathbb{F}_{p^k}$ dominates the complexity of the operations that occur in the lower degree subfields. In this section, we only analyze the cost of updating the functions f, g which are generally executed on the full extension field $\mathbb{F}_{p^k}$. Assume that the ratio of one squaring to one multiplication in the full extension field is set to be $\mathbf{S}_{p^k} = 0.8\mathbf{M}_{p^k}$ as commonly used value in the literature.

For a doubling (lines 1, 2), Algorithm 2 requires $2\mathbf{M}_{p^k} + 2\mathbf{S}_{p^k}$, while Algorithm 3 requires only $1\mathbf{M}_{p^k} + 2\mathbf{S}_{p^k}$ in order to update functions f and g. Likewise, an addition step requires $2\mathbf{M}_{p^k}$ in Algorithm 2 (lines 3, 4), while this step requires only $1\mathbf{M}_{p^k}$

Table 1 Comparison of the cost of updating f, g of algorithms

	Doubling	Doubling and addition
Algorithm 1 (Miller's algorithm [1,24])	$2\mathbf{S}_{p^k} + 3\mathbf{M}_{p^k} = 4.6\mathbf{M}_{p^k}$	$2\mathbf{S}_{p^k} + 5\mathbf{M}_{p^k} = 6.6\mathbf{M}_{p^k}$
Algorithm in [1] (*even* embedding degrees)	$1\mathbf{S}_{p^k} + 1\mathbf{M}_{p^k} = 1.8\mathbf{M}_{p^k}$	$1\mathbf{S}_{p^k} + 2\mathbf{M}_{p^k} = 2.8\mathbf{M}_{p^k}$
Algorithm 2	$2\mathbf{S}_{p^k} + 2\mathbf{M}_{p^k} = 3.6\mathbf{M}_{p^k}$	$2\mathbf{S}_{p^k} + 4\mathbf{M}_{p^k} = 5.6\mathbf{M}_{p^k}$
Algorithm 3	$2\mathbf{S}_{p^k} + 1\mathbf{M}_{p^k} = 2.6\mathbf{M}_{p^k}$	$2\mathbf{S}_{p^k} + 2\mathbf{M}_{p^k} = 3.6\mathbf{M}_{p^k}$ (line **3**)
		$2\mathbf{S}_{p^k} + 3\mathbf{M}_{p^k} = 4.6\mathbf{M}_{p^k}$ (line **4**)

"Doubling" is when algorithms deal with the bit "$r_i = 0$" and "Doubling and Addition" is when algorithms deal with the bit "$r_i = 1$"

by Algorithm 3.[3] Table 1 shows the number of operations needed in $\mathbb{F}_{p^k}$ for updating f, g in different algorithms.

Obviously, Algorithm 3 offers a better performance than Algorithm 2. From Table 1, for the *generic* case we can see that Algorithm 3 saves two full extension field multiplications when the bit $r_i = 0$ as compared to the original Miller algorithm (Algorithm 1). When the bit $r_i = 1$, Algorithm 3 saves two or three full extension field multiplications in comparison with Algorithm 1, depending on which case Algorithm 3 executes (i.e., line 3 or line 4).

In comparison with Arene et al.'s algorithm [1], Algorithm 3 requires one more squaring in the full extension field for each doubling step. However, as mentioned early Arene et al.'s algorithm can only be applied on Edwards curves with an *even* embedding degree k for Tate pairing computation, while our approach is *generic*. It can be applied to *any* (pairing-friendly) Edwards curve and for *both* the Weil and the Tate pairing.

Let $r = \sum_{i=0}^{l'-1} q_i 4^i$, with $q_i \in \{0, 1, 2, 3\}$ be the representation in radix-4 of the group order r. The refinements in [24] are described in radix 4. Their algorithm ([24, Algorithm3]) allows to eliminate some rational line functions from Eq. (3) during pairing computations. Their best results could be obtained by combining formula (1) and (2) of Theorem 2. In particular, they used the first formula for cases $q_i = 0, 1, 2$, and the second formula in the case $q_i = 3$. Table 2 compares Algorithm 3 and their algorithm. Because our algorithm works in radix 2, the number of operations in radix 4 could be counted by simply counting from Table 1. Notice that line 4 in Algorithm 3 (the most costly case in our algorithm) will never be performed in two consecutive iterations because the value of m will never be equal to 0 for two consecutive iterations (see Algorithm 3). Thus, when $q_i = 3$ the worst case of Algorithm 3 happens as follows: the first bit "1" will be computed by using line **4** (cost $2\mathbf{S}_{p^k} + 3\mathbf{M}_{p^k}$), and then value value of m is set to be 1; the second bit "1" will be computed by using line **3** (cost $2\mathbf{S}_{p^k} + 2\mathbf{M}_{p^k}$).

[3] Lines 3, 4 in Algorithm 3 combine both a doubling and an addition step.

Table 2 Comparison of our algorithm with the refinements in [24]

	Algorithm in [24]	Algorithm 3
$q_i = 0$	$5\mathbf{S}_{p^k} + 3\mathbf{M}_{p^k} = 7\mathbf{M}_{p^k}$	$4\mathbf{S}_{p^k} + 2\mathbf{M}_{p^k} = 5.2\mathbf{M}_{p^k}$
$q_i = 1$	$4\mathbf{S}_{p^k} + 7\mathbf{M}_{p^k} = 10.2\mathbf{M}_{p^k}$	$4\mathbf{S}_{p^k} + 3\mathbf{M}_{p^k} = 6.2\mathbf{M}_{p^k}$ (line **3**)
		$4\mathbf{S}_{p^k} + 4\mathbf{M}_{p^k} = 7.2\mathbf{M}_{p^k}$ (line **4**)
$q_i = 2$	$4\mathbf{S}_{p^k} + 7\mathbf{M}_{p^k} = 10.2\mathbf{M}_{p^k}$	$4\mathbf{S}_{p^k} + 3\mathbf{M}_{p^k} = 6.2\mathbf{M}_{p^k}$ (line **3**)
		$4\mathbf{S}_{p^k} + 4\mathbf{M}_{p^k} = 7.2\mathbf{M}_{p^k}$ (line **4**)
$q_i = 3$	$4\mathbf{S}_{p^k} + 10\mathbf{M}_{p^k} = 13.2\mathbf{M}_{p^k}$	$4\mathbf{S}_{p^k} + 4\mathbf{M}_{p^k} = 7.2\mathbf{M}_{p^k}$ (line **3**)
		$4\mathbf{S}_{p^k} + 5\mathbf{M}_{p^k} = 8.2\mathbf{M}_{p^k}$ (line **4**)

Table 2 shows that Algorithm 3 is *obviously faster* than the refinements of Miller's algorithm in [24] for all four cases. Our algorithm saves two operations (resp., at least 3, or 5 operations) occurring in the full extension field $\mathbb{F}_{p^k}$ for $q_i = 0$ (resp., $q_i = 1, 2$ or $q_i = 3$). When the embedding degree k gets larger, the complexity of these operations dominates the complexity of these operations occurring in $\mathbb{F}_p$, then our algorithm is approximately 25 % faster than Xu–Lin's algorithm.

5 Conclusion

In this paper, we proposed further refinements to Miller's algorithm over Edwards curves. Our refinements *significantly* improved the performance of Miller's algorithm in comparison with the original Miller algorithm and its previous refinements. Our method is *generic*, hence the proposed algorithm can be applied for computing pairings of any type over any pairing-friendly Edwards curve. This allows the use of Edwards curves with *odd* embedding degree, and is suitable for the computation of *optimal pairings*.

References

1. Arène, C., Lange, T., Naehrig, M., Ritzenthaler, C.: Faster computation of the Tate pairing. J. Number Theory **131**(5), 842–857 (2011)
2. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted Edwards curves. In: Proceedings of the Cryptology in Africa 1st International Conference on Progress in Cryptology. AFRICACRYPT'08, pp. 389–405. Springer, Berlin/Heidelberg (2008)
3. Boxall, J., El Mrabet, N., Laguillaumie, F., Le, D.-P.: A variant of Miller's formula and algorithm. In: Proceedings of the 4th International Conference on Pairing-Based Cryptography, Pairing'10, Springer, Berlin, Heidelberg, pp. 417–434 (2010)
4. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, pp. 213–229. Springer, Heidelberg (2001)
5. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient algorithms for pairing-based cryptosystems. In: CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology. Springer, London, UK, pp. 354–368 (2002)

6. Barreto, P.S., Galbraith, S.D., Héigeartaigh, C.Ó., Scott, M.: Efficient pairing computation on supersingular abelian varieties. Des. Codes Cryptogr. **42**(3), 239–271 (2007)
7. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: Proceedings of the Advances in Crypotology 13th International Conference on Theory and Application of Cryptology and Information Security. ASIACRYPT'07, pp. 29–50. Springer, Berlin, Heidelberg (2007)
8. Bernstein, D.J., Lange, T.: Inverted Edwards coordinates. In: Proceedings of the 17th International Conference on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes. AAECC'07, pp. 20–27. Springer, Berlin, Heidelberg (2007)
9. Bernstein, D.J., Lange, T.: A complete set of addition laws for incomplete Edwards curves. J. Number Theory **131**(5), 858–872 (2011)
10. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Proceedings of SAC 2005. Volume 3897 of LNCS, pp. 319–331. Springer, Heidelberg (2005)
11. Blake, I.F., Murty, V.K., Xu, G.: Refinements of Miller's algorithm for computing the Weil/Tate pairing. J. Algorithms **58**(2), 134–149 (2006)
12. Costello, C., Lange, T., Naehrig, M.: Faster pairing computations on curves with high-degree twists. In: Nguyen, P., Pointcheval, D. (eds.) Public Key Cryptography—PKC 2010 Volume 6056 of Lecture Notes in Computer Science, pp. 224–242. Springer, Berlin/Heidelberg (2010)
13. Das, M.P., Sarkar, P.: Pairing computation on twisted edwards form elliptic curves. In: Proceedings of the 2nd International Conference on Pairing-Based Cryptography, pp. 192–210. Pairing '08. Springer, Berlin, Heidelberg (2008)
14. Hess, F.: Pairing lattices. In: Proceedings of the 2nd International Conference on Pairing-Based Cryptography. Pairing '08, pp. 18–38. Springer, Berlin, Heidelberg (2008)
15. Hess, F., Smart, N.P., Vercauteren, F.: The Eta pairing revisited. IEEE Trans. Inf. Theory **52**, 4595–4602 (2006)
16. Hisil, H., Wong, K.K.-H., Carter, G., Dawson, E.: Twisted Edwards curves revisited. In: Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology. ASIACRYPT '08, pp. 326–343. Springer, Berlin, Heidelberg (2008)
17. Ionica, S., Joux, A.: Another approach to pairing computation in edwards coordinates. In: Progress in Cryptology—INDOCRYPT 2008. Lecture Notes in Computer Science, vol. 5365, pp. 400–413. Springer, Berlin/Heidelberg (2008)
18. Joux, A.: A one round protocol for tripartite Diffie-Hellman. In: ANTS-IV: Proceedings of the 4th International Symposium on Algorithmic Number Theory, pp. 385–394. Springer, Berlin (2000)
19. Le, D.-P., Liu, C.-L.: Refinements of Miller's algorithm over Weierstrass curves revisited. Comput. J. **54**(10), 1582–1591 (2011)
20. Le, D.-P., Tan, C.H.: Improved Miller's algorithm for computing pairings on Edwards curves. IEEE Trans. Comput. **63**(10), 2626–2632 (2014)
21. Miller, V.S.: Short programs for functions on curves. IBM Thomas J. Watson Research Center, New York (1986)
22. Miller, V.S.: The Weil pairing, and its efficient calculation. J. Cryptol. **17**(4), 235–261 (2004)
23. Vercauteren, F.: Optimal pairings. IEEE Trans. Inf. Theory **56**(1), 455–461 (2010)
24. Xu, L., Lin, D.: Refinement of Miller's algorithm over Edwards curves. In: Pieprzyk, J. (ed.) CT-RSA. Lecture Notes in Computer Science, vol. 5985, pp. 106–118. Springer, Berlin (2010)