Encapsulated Scalar Multiplications and Line Functions in the Computation of Tate Pairing*

Rongquan Feng** and Hongfeng Wu

LMAM, School of Math. Sciences, Peking University, Beijing 100871, P.R. China
fengrq@math.pku.edu.cn, wuhf@math.pku.edu.cn

Abstract. The efficient computation of the Tate pairing is a crucial factor to realize cryptographic applications practically. To compute the Tate pairing, two kinds of costs on the scalar multiplications and Miller's line functions of elliptic curves should be considered. In the present paper, encapsulated scalar multiplications and line functions are discussed. Some simplified formulas and improved algorithms to compute f_{3T}, f_{4T}, $f_{2T\pm P}$, f_{6T}, $f_{3T\pm P}$ and $f_{4T\pm P}$ etc., are presented from given points T and P on the elliptic curve.

Keywords: Elliptic curve, scalar multiplication, line function, Tate pairing, Miller's path.

1 Introduction

With the discovery of the identity-based encryption scheme based on the Weil pairing by Boneh and Franklin [4], cryptographic protocols based on the Weil and Tate pairings on elliptic curves have attracted much attention in recent years. Many cryptographic schemes based on the Weil or on the Tate pairing have been introduced. The readers are referred to [11] for a survey. In most of these pairing-based cryptographic schemes, the Tate pairing is the essential tool. The pairing computations are often the bottleneck to realize the cryptographic applications in practice. Therefore efficient computation of the Tate pairing is a crucial factor to realize cryptographic applications practically.

Miller [20,21] provided an algorithm to compute the Weil/Tate pairing. The Miller's algorithm itself consists of two parts: the Miller's function f_{rP} and a final exponentiation. The main part of the computation for the Tate pairing is calculating $f_{rP}(Q)$. In the computation of the Miller's function $f_{(m+n)P}(Q)$, one needs to perform a conditional scalar multiplication and compute the line functions $l_{mP,nP}(Q)$ and $l_{(m+n)P}(Q)$ from points mP and nP on the elliptic curve. The total cost of the computation of the Tate pairing is the sum of the cost of scalar multiplications and that of the computation of line functions $l_{mP,nP}(Q)$ and $l_{(m+n)P}(Q)$. So a good algorithm must think over these two kinds of costs

* Supported by the NSF of China (10571005, 60473019), by 863 Project (No. 2006AA01Z434) and by NKBRPC (2004CB318000).

** Corresponding author.

J.-Y. Cai, S.B. Cooper, and H. Zhu (Eds.): TAMC 2007, LNCS 4484, pp. 159–170, 2007.

simultaneously. In the first, we need to find more efficient method to compute the scalar multiplication, at the same time we need to modify the algorithm so that it can be serviced to make the computation of line functions. Secondly, we need other strategies to change and simplify the formula of $f_{(m+n)P}$ so that the good algorithm of scalar multiplications can be used to the computation of $f_{(m+n)P}$. In the computation of the Tate pairing, we often need to compute the following Miller's functions f_{4T}, f_{3T}, $f_{iT\pm P}$, f_{2^kT}, f_{3^kT}, f_{6T} etc. In this paper, efficient algorithms to compute some of these Miller's functions are presented. We propose a useful fact and use it to simplify f_{m+n}. The simplified formula of f_{m+n} and new point multiplication algorithms make our algorithms more efficient.

This paper is organized as follows. Some essential concepts to discuss pairings are reviewed in Section 2. In Section 3, some efficient algorithms to compute different Miller's paths are described. An example using those results is given in Section 4. And finally conclusions are proposed in Section 5.

2 Preliminaries

In this section, the basic arithmetic of elliptic curves, the Tate pairing and Miller's algorithm are described briefly. The readers are referred to [1] and [25] for more details. Throughout this paper, the base field K is always assumed to be the finite field $\mathbb{F}_q$, where $q = p^m$ with $p > 3$, and $\overline{K}$ is the algebraic closure of K.

An elliptic curve E over K is a curve that is given by an equation of the form

$$y^2 = x^3 + ax + b, \tag{1}$$

where $a, b \in K$ and $4a^3 + 27b \neq 0$. Let $E(K)$ denote the set of points $(x, y) \in K^2$ which satisfy the equation (1), along with a "point at infinity", denoted by $\mathcal{O}$. For any positive integer k, $F = \mathbb{F}_{q^k}$ is an extension field of K. Then $E(F)$ denotes the set of $(x, y) \in F^2$ that satisfy (1), along with $\mathcal{O}$. There is an abelian group structure in $E(K)$. The addition formulas for affine coordinates are the followings. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two elements of $E(K)$ that are different from $\mathcal{O}$. Then the addition $P_3 = P_1 + P_2 = (x_3, y_3)$ is defined by $x_3 = \lambda_{P_1,P_2}^2 - x_1 - x_2$ and $y_3 = \lambda_{P_1,P_2}(x_1 - x_3) - y_1$, where $\lambda_{P_1,P_2} = (y_2 - y_1)/(x_2 - x_1)$ is the slope of the line through P_1 and P_2 for $P_1 \neq \pm P_2$ and $\lambda_{P_1,P_2} = (3x_1^2 + a)/(2y_1)$ is the slope of the tangent line at P_1 for $P_1 = P_2$.

Let ℓ be a positive integer and let $E[\ell]$ (resp. $E(\mathbb{F}_q)[\ell]$) be the set of points $P \in E(\overline{\mathbb{F}_q})$ (resp. $P \in E(\mathbb{F}_q)$) satisfying $\ell P = \mathcal{O}$. Let P be a point on $E(\mathbb{F}_q)$ of order r, the point P, or the cyclic subgroup $\langle P \rangle$, or the integer r is said to have *embedding degree* k for some positive integer k if $r \mid q^k - 1$ and $r \nmid q^s - 1$ for any $0 < s < k$. The group $E(\mathbb{F}_q)$ is (isomorphic to) a subgroup of $E(\mathbb{F}_{q^k})$. Let $P \in E(\mathbb{F}_q)$ be a point of order r such that P has embedding degree k. Then $E(\mathbb{F}_{q^k})$ contains a point Q of the same order r but linearly independent with P (see [1]).

Definition 1. *Let r be a positive integer coprime to q and let k be the embedding degree to r. The Tate pairing $\tau_r\langle\cdot,\cdot\rangle$ is a map $\tau_r\langle\cdot,\cdot\rangle : E(\mathbb{F}_q)[r]\times E(\mathbb{F}_{q^k})[r] \to \mathbb{F}^*_{q^k}$ defined by $\tau_r(P,Q) = f_{rP}(D_Q)^{(q^k-1)/r}$ for any $P \in E(\mathbb{F}_q)[r]$ and $Q \in E(\mathbb{F}_{q^k})[r]$, where f_{rP} is a rational function satisfying $(f_{rP}) = r(P) - r(\mathcal{O})$, and $D_Q \sim (Q) - (\mathcal{O})$ such that (f_P) and D_Q have disjoint supports.*

From [2], we know that $\tau_r(P,Q)$ can be reduced to $\tau_r(P,Q) = f_{rP}(Q)^{(q^k-1)/r}$ when $r \nmid (p-1)$, $Q \neq \mathcal{O}$, and $k > 1$. Noting that in most cryptographic primitives, r is set to be a prime such that $r \mid \sharp E(\mathbb{F}_q)$. Furthermore, in practice, r is at least larger than 160 bits.

In order to compute the Tate pairing $\tau_r(P,Q)$, we need to find the function f_P and then evaluate its value at Q. The algorithm, proposed by Miller [20], and then called Miller's algorithm, can be used to compute the Tate paring. Denoted by $l_{U,V}$ the equation of the line through points $U, V \in E(\mathbb{F}_q)$. Naturally, if $U = V$, then $l_{U,V}$ is the equation of the tangent line at U, and if either U or V is the point at infinity $\mathcal{O}$, then $l_{U,V}$ represents the vertical line through the other point. Furthermore, for simplicity, we write l_U instead of $l_{U,-U}$. For a point P on elliptic curve E, define a Miller's function with parameter $n \in \mathbb{N}$ to be a rational function f_{nP} (or simply f_n) on E such that $(f_n) = n(P) - (nP) - (n-1)(\mathcal{O})$.

Theorem 1 ([1] Miller's formula). *Let $P \in E(\mathbb{F}_q)$, n be an integer and let f_n be the function with divisor $(f_n) = n(P) - (nP) - (n-1)(\mathcal{O})$. Then for any $m, n \in \mathbb{Z}$,*

$$f_{m+n}(Q) = f_m(Q) \cdot f_n(Q) \cdot \frac{l_{mP,nP}(Q)}{l_{(m+n)P}(Q)}.$$

We can use the Miller's algorithm to compute f_n and then evaluate the Tate pairing. The standard double-and-add method to compute the Tate pairing is the following algorithm.

Algorithm 1. *Miller's algorithm:*

Input: $t = \log r$, $r = (r_t = 1, r_{t-1}, \ldots, r_0)_2$, P, Q.
Output: $f_{rP}(Q)^{(q^k-1)/r}$.

1: $T = P$, $f = 1$
2: *For* $i = t-1$ *downto* 0 *do*
3: $f \leftarrow f^2 \cdot l_{T,T}(Q)/l_{2T}(Q)$ *and* $T \leftarrow 2T$;
4: *if* $r_i = 1$, *then* $f \leftarrow f \cdot l_{T,P}(Q)/l_{T+P}(Q)$ *and* $T \leftarrow T + P$
5: Return $f^{(q^k-1)/r}$

Definition 2. *Let $P \in E(\mathbb{F}_q)$ and $Q \in E(\mathbb{F}_{q^k})$, a Miller's path about Q from $(n_1P, n_2P, \cdots, n_sP)$ to $(n_1+n_2+\cdots+n_s)P$ is an algorithm* **A** *which can output $(n_1+n_2+\cdots+n_s)P$ and $f_{(n_1+n_2+\cdots+n_s)}(Q)$ when input $(n_1P, n_2P, \cdots, n_sP)$ and $f_{n_1}, f_{n_2}, \cdots, f_{n_s}$. When there is no confusion, this algorithm* **A** *is said to be a Miller's path to $(n_1P, n_2P, \cdots, n_sP)$.*

The computational cost (timing) of scalar multiplications and Tate pairings on elliptic curve operations depend on the cost of the arithmetic operations that have to be performed in the underlying field. In general, among these arithmetics, a field squaring, a field multiplication and a field inversion are more expensive than other field arithmetics, such as a field addition and a field subtraction. So we only take into account the cost of inversion, multiplication, and squaring in the field $\mathbb{F}_q$, which we denote by I, M and S, respectively, while in the extension field $\mathbb{F}_{q^k}$, those costs are denoted by I_k, M_k and S_k, respectively. Generally it is assumed that $1S = 0.8M$, $1M_k = k^2M$ and $1I_k = k^2M + I$. Also the multiplication between elements in $\mathbb{F}_q^*$ and $\mathbb{F}_{q^k}^*$ costs kM.

3 Computation of Miller's Paths

In the process of computing Tate pairings, we often need to compute $(f_{mP+nT}, mP+nT)$ from (f_P, P) and (f_T, T). A Miller's path is to compute $mP+nT$ and f_{mP+nT} at the same time. Different Miller's paths have different computation costs. A main problem is to find an optimized Miller's path so that we can quicken the computation of Tate pairings.

Throughout this section, we assume that $P \in E(\mathbb{F}_q)$ and $Q \in E(\mathbb{F}_{q^k})$. Moreover, though we don't use the denominators discarded method [1] in here, our strategies can also be used in those algorithms there.

3.1 Miller's Path to 4T

In this subsection, an improved method for obtaining $(f_{4T}, 4T)$ from (f_T, T) is given. Firstly, some facts about elliptic curves will be given from which one can simplify the computation of f_{4T}.

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be points on E, Set $P_1 + P_2 = P_3 = (x_3, y_3)$. Let $l_{P_1,P_2} : y - \lambda_{P_1,P_2}(x - x_1) - y_1 = 0$ be the equation of the line through P_1 and P_2 and $l_{P_1+P_2} : x - x_3 = 0$ be the vertical line through the point $P_1 + P_2$. If $P_1 = P_2$ then l_{P_1} is the tangent line at P_1, and if $P_1 + P_2 = \mathcal{O}$ then we take $l_{P_1+P_2} = 1$.

Lemma 1. *For $P_1, P_2 \in E(\mathbb{F}_q)$, let $P_1 + P_2 = P_3$. Then we have $l_{P_1,P_2} \cdot l_{-P_1,-P_2} = l_{P_1} \cdot l_{P_2} \cdot l_{P_3}$, i.e.,*

$$(y - \lambda_{P_1,P_2}(x - x_1) - y_1)(y + \lambda_{P_1,P_2}(x - x_1) + y_1) = (x - x_1)(x - x_2)(x - x_3).$$

Proof. The divisor of the function l_{P_1,P_2} is $(l_{P_1,P_2}) = (P_1)+(P_2)+(-P_3)-3(\mathcal{O})$. Since $-P_1 = (x_1, -y_1)$, we have $(y + \lambda_{P_1,P_2}(x - x_1) + y_1) = (-P_1) + (-P_2) + (P_3) - 3(\mathcal{O})$. Thus

$$\begin{aligned}
&\operatorname{div}((y - \lambda_{P_1,P_2}(x - x_1) - y_1)(y + \lambda_{P_1,P_2}(x - x_1) + y_1)) \\
&= (P_1) + (P_2) + (-P_3) - 3(\mathcal{O}) + (-P_1) + (-P_2) + (P_3) - 3(\mathcal{O}) \\
&= (P_1) + (-P_1) - 2(\mathcal{O}) + (P_2) + (-P_2) - 2(\mathcal{O}) + (P_3) + (-P_3) - 2(\mathcal{O}) \\
&= \operatorname{div}(x - x_1) + \operatorname{div}(x - x_2) + \operatorname{div}(x - x_3) \\
&= \operatorname{div}((x - x_1)(x - x_2)(x - x_3)).
\end{aligned}$$

From $\mathrm{div}(f) = 0$ if and only if f is a constant function, we have that

$$(y - \lambda_{P_1,P_2}(x - x_1) - y_1)(y + \lambda_{P_1,P_2}(x - x_1) + y_1) = c \cdot (x - x_1)(x - x_2)(x - x_3)$$

for some constant $c \in K$. But since the coefficient of x^3 is 1 in both sides we have $c = 1$. □

Remark 1. From the lemma 1, we know $f_{m-n}(Q) = \dfrac{f_m}{f_n}(Q) \cdot \dfrac{l_{mP}}{l_{-mP,nP}}(Q)$, Thus we have $f_{2m-1}(Q) = f_m^2(Q) \cdot \dfrac{l_{mP,mP}}{l_{-2mP,P}}(Q)$.

We now describe an improved method for obtaining $(f_{4T}, 4T)$ from (f_T, T). Note that for any points T and S, we have $l_{-T,-S}(Q) = l_{T,S}(-Q)$. By Lemma 1, we have

$$f_{4T} = \left(f_T^2 \cdot \frac{l_{T,T}}{l_{2T}}\right)^2 \cdot \frac{l_{2T,2T}}{l_{4T}} = f_T^4 \cdot \frac{l_{T,T}^2}{l_{2T}^2} \cdot \frac{l_{2T,2T}}{l_{4T}} = f_T^4 \cdot \frac{l_{T,T}^2}{l_{-2T,-2T}}.$$

Let $T = (x_1, y_1)$, $2T = (x_2, y_2)$ and $4T = (x_4, y_4)$. Then $x_2 = \lambda_{T,T}^2 - 2x_1$, $y_2 = \lambda_{T,T}(x_1 - x_2) - y_1$; $x_4 = \lambda_{2T,2T}^2 - 2x_2$ and $y_4 = \lambda_{2T,2T}(x_2 - x_4) - y_2$. Set $Q = (x, y)$. We have

$$f_{4T}(Q) = f_T^4 \cdot \frac{l_{T,T}^2}{l_{-2T,-2T}}(Q) = \frac{f_T^4 \cdot l_{T,T}^2(Q)}{l_{2T,2T}(-Q)} = f_T^4 \cdot \frac{[y - y_1 - \lambda_{T,T}(x - x_1)]^2}{y + y_2 + \lambda_{2T,2T}(x + x_2)}.$$

Furthermore, let λ be defined as $\lambda = 3x_1^2 + a$. Then we have

$$\frac{1}{2y_2} = \frac{(2y_1)^3}{2\lambda(3x_1 \cdot (2y_1)^2 - \lambda^2) - (2y_1)^4}.$$

Thus

$$\lambda_{2T,2T} = (3x_2^2 + a) \cdot \frac{(2y_1)^4}{2y_1[2\lambda(3x_1 \cdot (2y_1)^2 - \lambda^2) - (2y_1)^4]};$$
$$\lambda_{T,T} = (3x_1^2 + a) \cdot \frac{2\lambda(3x_1 \cdot (2y_1)^2 - \lambda^2) - (2y_1)^4}{2y_1[2\lambda(3x_1 \cdot (2y_1)^2 - \lambda^2) - (2y_1)^4]}.$$

So we have the following algorithm to compute $4T$ and f_{4T} from T and f_T.

In the above algorithm, we need $I + 7S + 9M$ to compute λ_1, λ_2 and (x_4, y_4). It is cheaper than two doubling method which cost $2I + 4S + 4M$ for in the general finite field $I/M \geq 10$. Also it is better than the algorithm in [6] where the cost is $I + 9S + 9M$. Steps 7 and 8 cost $I_k + 2S_k + 2M_k + 2kM$ when $Q = (x, y)$, $x, y \in \mathbb{F}_{q^k}$. For general algorithm, the cost is $2I_k + S_k + 2M_k + 2kM$ (see [21]). So this algorithm is better than known algorithms not only for $4T$, but also for f_{4T}.

Algorithm 2. *(Path to 4T algorithm):*

———————————————————————————-

Input: $T = (x_1, y_1), Q = (x, y),\ f_T$.
Output: $f_{4T}(Q),\ 2T,\ 4T = (x_4, y_4)$.

———————————————————————————-

1: $t_1 = 3x_1^2 + a;\ t_2 = 2y_1; t_3 = (t_2^2)^2;\ t_4 = x_1 \cdot t_3;$
2: $t_5 = 2t_1 \cdot (3t_4 - t_1^2) - t_3;$
3: $t_6 = (t_2 \cdot t_5)^{-1};$
4: $\lambda_1 = t_1 \cdot t_5 \cdot t_6;\ x_2 = \lambda_1^2 - 2x_1;\ y_2 = \lambda_1(x_1 - x_2) - y_1;$
5: $\lambda_2 = (3x_2^2 + a) \cdot t_3 \cdot t_6;\ x_4 = \lambda_2^2 - 2x_2;\ y_4 = \lambda_2(x_2 - x_4) - y_2;$
6: $f_1 = y - y_1 - \lambda_1(x - x_1);\ f_2 = y + y_2 + \lambda_2(x + x_2);$
7: $f_{4T}(Q) = (f_T(Q)^2 \cdot f_1(Q))^2 \cdot f_2(Q)^{-1};$
8: Return $f_{4T}(Q),\ 2T,\ 4T = (x_4, y_4)$.

———————————————————————————-

3.2 Miller's Path to $2T \pm P$

In this subsection, we will describe the efficient Miller's path to $(f_{2T+P}, 2T + P)$ and $(f_{2T-P}, 2T - P)$.

Miller's Path to $2T + P$. We first give an efficient Miller's path to $(f_{2T+P}, 2T + P)$ from (f_T, T) and (f_P, P) when $T \neq P$.

First noting that

$$f_{2T+P} = f_{T+P} \cdot \frac{f_T \cdot l_{T+P,T}}{l_{2T+P}} = \frac{f_T \cdot f_P \cdot l_{T,P}}{l_{T+P}} \cdot \frac{f_T \cdot l_{T+P,T}}{l_{2T+P}} = \frac{f_T^2 \cdot f_P}{l_{2T+P}} \cdot \frac{l_{T,P} \cdot l_{T+P,T}}{l_{T+P}}.$$

Set $T = (x_1, y_1)$, $P = (x_2, y_2)$, $T + P = (x_3, y_3)$ and $2T + P = (x_4, y_4)$. From [12] we can replace $(l_{T,P} \cdot l_{T+P,T})/l_{T+P}$ by the following parabolas formula:

$$\frac{l_{T,P} \cdot l_{T+P,T}}{l_{T+P}} = (x - x_1)(x + x_1 + x_3 + \lambda_{T,P}\lambda_{T,T+P}) - (\lambda_{T,P} + \lambda_{T,T+P})(y - y_1).$$

Furthermore, $\lambda_{T,T+P}$ can be expanded as follows:

$$\begin{aligned}\lambda_{T,T+P} = \frac{y_3 - y_1}{x_3 - x_1} &= \frac{(x_1 - x_3)\lambda_{T,P} - 2y_1}{\lambda_{T,P}^2 - 2x_1 - x_2} \\ &= \frac{2y_1(x_2 - x_1)^3 - (y_2 - y_1)\left[(x_2 - x_1)^2(2x_1 + x_2) - (y_2 - y_1)^2\right]}{(x_2 - x_1)\left[(x_2 - x_1)^2(2x_1 + x_2) - (y_2 - y_1)^2\right]}.\end{aligned}$$

Since

$$\lambda_{T,P} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{(y_2 - y_1)\left[(x_2 - x_1)^2(2x_1 + x_2) - (y_2 - y_1)^2\right]}{(x_2 - x_1)\left[(x_2 - x_1)^2(2x_1 + x_2) - (y_2 - y_1)^2\right]}.$$

So we need only to compute one inversion $\left\{(x_2 - x_1)\left[(x_2 - x_1)^2(2x_1 + x_2) - (y_2 - y_1)^2\right]\right\}^{-1}$ in order to compute $\lambda_{T,P}$ and $\lambda_{T,T+P}$ simultaneously. Thus we have the following formulas.

$$\begin{aligned}
\lambda_{T,P} &= \left\{(x_2-x_1)\left[(x_2-x_1)^2(2x_1+x_2)-(y_2-y_1)^2\right]\right\}^{-1} \\
&\quad \cdot (y_2-y_1)\left[(x_2-x_1)^2(2x_1+x_2)-(y_2-y_1)^2\right]; \\
\lambda_{T,T+P} &= \left\{(x_2-x_1)\left[(x_2-x_1)^2(2x_1+x_2)-(y_2-y_1)^2\right]\right\}^{-1} \cdot 2y_1(x_2-x_1)^3-\lambda_{T,P}; \\
x_4 &= (\lambda_{T,T+P}-\lambda_{T,P})(\lambda_{T,T+P}+\lambda_{T,P})+x_2; \\
y_4 &= (x_1-x_4)\lambda_{T,T+P}-y_1.
\end{aligned}$$

Since $x_3 = \lambda_{T,P}^2 - x_1 - x_2$, the new parabolas formula is

$$\frac{l_{T,P}\cdot l_{T+P,P}}{l_{T+P}} = (x-x_1)(\lambda_{T,P}(\lambda_{T,P}+\lambda_{T,T+P})-x_2+x)-(\lambda_{T,P}+\lambda_{T,T+P})(y-y_1).$$

This procedure is described by the following algorithm.

Algorithm 3. *(Path to $2T+P$ algorithm):*

Input: $T=(x_1,y_1)$, $P=(x_2,y_2)$, f_T, f_P *and* $Q=(x,y)$
Output: $f_{2T+P}(Q)$ *and* $2T+P=(x_4,y_4)$.

1: $t_1=(x_2-x_1)^2(2x_1+x_2)-(y_2-y_1)^2$,$t_2=(x_2-x_1)t_1$,$t_3=t_2^{-1}$;
2: $\lambda_1=(y_2-y_1)t_1t_3$, $\lambda_2=t_3\cdot 2y_1(x_2-x_1)^2(x_2-x_1)-\lambda_1$;
3: $x_4=(\lambda_2-\lambda_1)(\lambda_2+\lambda_1)+x_2$;
4: $y_4=(x_1-x_4)\lambda_2-y_1$;
5: $f_{2T+P}(Q)=\dfrac{f_T^2\cdot f_P}{l_{2T+P}}\cdot[(x-x_1)(\lambda_1(\lambda_1+\lambda_2)-x_2+x)-(\lambda_1+\lambda_2)(y-y_1)](Q)$
6: return $f_{2T+P}(Q)$, $2T+P$.

In Algorithm 2, we require $1I+2S+10M$ to compute $2T+P$ and $x_1+x_3+\lambda_{T,P}\lambda_{T,T+P}$, while in [12], $2I+2S+4M$ times are required to compute them. We save $1I-6M$ field computations.

Miller's Path to $2T-P$. Now we describe an efficient Miller's path to $(f_{2T-P}, 2T-P)$ from (f_T,T) and (f_P,P) when $T\neq P$. We can use $2T+(-P)$ to get f_{2T-P} by Algorithm 3, but here we describe a direct path to f_{2T-P}.

By Remark 1 in section 3.1 we know

$$f_{2T-P}(Q) = f_T^2(Q)\cdot\frac{l_{T,T}}{l_{-2T,P}}(Q).$$

Let $P=(x_P,y_P)$,$T=(x_T,y_T)$, $2T=(x_{2T},y_{2T})$ and $2T-P=(x_{2T-P},y_{2T-P})$. Then $x_{2T}=\lambda_{T,T}^2-2x_T$, $y_{2T}=\lambda_{T,T}(x_T-x_{2T})-y_T$; $x_{2T-P}=\lambda_{2T,-P}^2-x_{2T}-x_P$ and $y_{2T-P}=\lambda_{2T,-P}(x_P-x_{2T-P})+y_P$. Set $Q=(x,y)$. We have

$$f_{2T-P}(Q) = f_T^2\cdot\frac{l_{T,T}}{l_{-2T,P}}(Q) = f_T^2(Q)\cdot\frac{y-y_T-\lambda_{T,T}(x-x_T)}{y-y_P+\lambda_{2T,-P}(x-x_P)}.$$

Furthermore, let λ be defined as $\lambda = 3x_T^2 + a$. Then we have

$$\lambda_{T,T} = \frac{(3x_T^2 + a) \cdot [\lambda^2 - (2x_T + x_P)(2y_T)^2]}{2y_T[\lambda^2 - (2x_T + x_P)(2y_T)^2]};$$
$$\lambda_{2T,-P} = \frac{(2y_T)^3(y_{2T} + y_P)}{2y_T[\lambda^2 - (2x_T + x_P)(2y_T)^2]}.$$

Therefore, when $Q = (x, y)$ and $x \in \mathbb{F}_{q^k}$, $y \in \mathbb{F}_{q^k}$, to complete the Miller's path $(f_{2T-P}, 2T - P)$ from (f_T, T) and (f_P, P) we need only $1I + 5S + (2k + 9)M + I_k + S_k + 2M_k$. However, we need $1I + 2S + (k + 10)M + I_k + S_k + 4M_k$ when use the Algorithm 3 to compute $(f_{2T-P}, 2T - P)$.

3.3 Miller's Path to $3T$

In this subsection, a Miller's path to $(f_{3T}, 3T)$ from (f_T, T) is given. By the Miller's formula we have

$$f_{3T} = f_T^3 \cdot \frac{l_{T,T}}{l_{2T}} \cdot \frac{l_{T,2T}}{l_{3T}}.$$

Let $T = (x_1, y_1)$, $2T = (x_2, y_2)$, and $3T = (x_3, y_3)$. Then

$$\frac{l_{T,T} \cdot l_{T,2T}}{l_{2T}} = (x - x_1)(x + x_1 + x_2 + \lambda_{T,T}\lambda_{T,2T}) - (\lambda_{T,T} + \lambda_{T,2T})(y - y_1).$$

By $x_2 = \lambda_{T,T}^2 - 2x_1$, we have the following parabolas formula

$$\frac{l_{T,T} \cdot l_{T,2T}}{l_{2T}} = (x - x_1)(x + \lambda_{T,T}^2 - x_1 + \lambda_{T,T}\lambda_{T,2T}) - (\lambda_{T,T} + \lambda_{T,2T})(y - y_1).$$

Noting that $x_3 = (\lambda_{T,2T} - \lambda_{T,T})(\lambda_{T,2T} + \lambda_{T,T}) + x_1$, we need only $\lambda_{T,T}$, $\lambda_{T,2T}$ and $3T$ to compute $(f_{3T}, 3T)$. From

$$\lambda_{T,2T} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\lambda_{T,T}(x_1 - (\lambda_{T,T}^2 - 2x_1)) - 2y_1}{(\lambda_{T,T}^2 - 2x_1) - x_1} = \frac{2y_1}{3x_1 - \lambda_{T,T}^2} - \lambda_{T,T}$$
$$= \frac{(2y_1)^3}{(2y_1)^2(3x_1) - (3x_1^2 + a)^2} - \lambda_{T,T}$$

and

$$\lambda_{T,T} = \frac{3x_1^2 + a^2}{2y_1} = \frac{3x_1^2 + a^2}{2y_1} \cdot \frac{(2y_1)^2(3x_1) - (3x_1^2 + a)^2}{(2y_1)^2(3x_1) - (3x_1^2 + a)^2},$$

we have the follow algorithm:

In step 5 of Algorithm 4, $I_k + S_k + 4M_k + (k+1)M$ costs are needed to compute $f_{3T}(Q)$. However the general double-add algorithm needs $I_k + S_k + 7M_k + 2kM$. Our algorithm saves $3M_k + (k-1)M$ field computations.

Algorithm 4. *(Path to 3T algorithm):*

Input: $T = (x_1, y_1)$, f_T, $Q = (x, y)$.
Output: $f_{3T}(Q)$ *and* $3T = (x_3, y_3)$.
1: $t_1 = (2y_1)^2$, $t_2 = t_1^2$, $t_3 = 3x_1^2 + a$;
2: $t_4 = 3x_1 \cdot t_1 - t_3$, $t_5 = (2y_1 \cdot t_4)^{-1}$, $\lambda_1 = t_3 t_4 t_5$, $\lambda_2 = t_2 t_5 - \lambda_1$;
3: $x_3 = (\lambda_2 - \lambda_1)(\lambda_2 + \lambda_1) + x_1$;
4: $y_3 = (x_1 - x_3)\lambda_2 - y_1$;
5: $f_{3T}(Q) = \dfrac{f_T^3(Q)}{x - x_3} \cdot [(x - x_1)(\lambda_1(\lambda_1 + \lambda_2) - x_1 + x) - (\lambda_1 + \lambda_2)(y - y_1)](Q)$;
6: Return $f_{3T}(Q)$, $3T$.

3.4 Miller's Path to $6T$

In this subsection, the Miller's path to $(f_{6T}, 6T)$ from (f_T, T) is considered. In the first we see the Miller's path $[(f_T, T) \to (f_{2T}, 2T)$ and $(f_T, T) \to (f_{4T}, 4T)] \to (f_{2T+4T}, 2T + 4T)$. By Miller's formula and the results in Section 3.1, we have

$$f_{6T} = f_{4T} \cdot f_{2T} \cdot \frac{l_{2T,4T}}{l_{6T}} = f_T^6 \cdot \frac{l_{T,T}^2}{l_{-2T,-2T}} \cdot \frac{l_{T,T}}{l_{2T}} \cdot \frac{l_{2T,4T}}{l_{6T}} = f_T^6 \cdot \frac{l_{T,T}^2}{l_{-2T,-2T}} \cdot \frac{l_{T,T} \cdot l_{2T,4T} \cdot l_{4T}}{l_{2T} \cdot l_{6T} \cdot l_{4T}}.$$

So by Lemma 1,

$$f_{6T} = f_T^6 \cdot \frac{l_{T,T}^3}{l_{-2T,-2T}} \cdot \frac{l_{2T,4T} \cdot l_{4T}}{l_{2T,4T} \cdot l_{-2T,-4T}} = f_T^6 \cdot \frac{l_{T,T}^3}{l_{-2T,-2T}} \cdot \frac{l_{4T}}{l_{-2T,-4T}}.$$

Let $T = (x_1, y_1)$, $2T = (x_2, y_2)$, $4T = (x_4, y_4)$ and $Q = (x, y)$. Then

$$\begin{aligned} f_{6T}(Q) &= (f_T^2 l_{T,T})^3 \cdot \frac{l_{4T}}{l_{-2T,-2T} \cdot l_{-2T,-4T}}(Q) \\ &= \frac{(f_T^2 l_{T,T})^3 \cdot (x - x_4)}{[y + y_2 + \lambda_{2T,2T}(x + x_2)] \cdot [y + y_2 + \lambda_{2T,4T}(x + x_2)]}. \end{aligned}$$

Secondly, there is another way to $(f_{6T}, 6T)$ from (f_T, T) as $(f_T, T) \to (f_{3T}, 3T) \to (f_{3T+3T}, 6T)$. Similarly, we have

$$\begin{aligned} f_{6T}(Q) &= f_{3T}^2 \cdot \frac{l_{3T,3T}}{l_{6T}}(Q) \\ &= f_T^6 \cdot \frac{[(x - x_1)(\lambda_{T,T}(\lambda_{T,T} + \lambda_{T,2T}) - x_1 + x) - (\lambda_{T,T} + \lambda_{T,2T})(y - y_1)]^2}{l_{3T}^2} \cdot \frac{l_{3T,3T}}{l_{6T}}(Q) \\ &= f_T^6 \cdot \frac{[(x - x_2)(\lambda_{T,T}(\lambda_{T,T} + \lambda_{T,2T}) - x_2 + x) - (\lambda_{T,T} + \lambda_{T,2T})(y - y_2)]^2}{l_{-3T,-3T}(Q)}. \end{aligned}$$

By comparing with their costs, the second way $(f_T, T) \to (f_{3T}, 3T) \to (f_{3T+3T}, 6T)$ is more efficient to compute $(f_{6T}, 6T)$. From this way, an algorithm to compute f_{6T} and $6T$ can be gotten.

3.5 Miller's Path to $iT \pm P$ for $i = 3, 4, 6$.

In this subsection, we think about the Miller's path to $iT \pm P$ from (f_T, T) and (f_P, P), where $i = 3$, 4 and 6. Firstly, let us see an optimized one to $(f_{3T+P}, 3T + P)$.

There are following four ways to get the Miller's path to $3T + P$ from (f_T, T) and (f_P, P). The first is as $(f_T, T) \to (f_{3T}, 3T) \to (f_{3T+P}, 3T + P)$. The second is as $[(f_T, T) \to (f_{2T}, 2T)$ and$(f_P, P) \to (f_{P+T}, P+T)] \to (f_{2T+(T+P)}, 2T+(T+P))$. The third is as $(f_T, T) \to (f_{2T+P}, 2T+P) \to (f_{(2T+P)+T}, (2T+P)+T)$, and the fourth is as $(f_T, T) \to (f_{T+P}, T+P) \to (f_{2T+(T+P)}, 2T+(P+T)$. Comparing with the costs of these four ways we know that the second way is more efficient than the others. Therefore, we have $f_{3T+P}(Q) = f_T^3 \frac{l_{T,T} \cdot l_{T,P}}{l_{-2T,-T-P}}(Q)$. The details are omitted.

Similarly, for the Miller's paths to $4T + P$, We have the ways $(f_T, T) \to (f_{4T}, 4T) \to (f_{4T+P}, 4T + P)$ and $(f_T, T) \to (f_{2T}, 2T) \to (f_{2(2T)+P}, 2(2T) + P)$. Comparing with the costs of these 2 ways we know that the first way is more efficient than the second. Similarly, for the Miller's paths to $6T + P$, the way $(f_T, T) \to (f_{3T}, 3T) \to (f_{2(3T)+P,2(3T)+P})$ is more efficient. From these results, algorithms to compute f_{3T+P}, f_{4T+p} and f_{6T+P} can be obtained.

The Miller's path to $iT - P$ are used to the algorithms using addition-subtraction chains. Under conditions, direct compute f_{iT-P} is more efficient. For example, for the path to $3T - P$ we have $f_{3T-P}(Q) = f_T^3[\frac{l_{T,T} l_{T,2T}}{l_{2T}}] \frac{1}{l_{-3T,P}}(Q)$. For the path to $4T - P$ we have $f_{4T-P}(Q) = f_{4T}(Q) \cdot \frac{l_{4T}}{l_{-4T,P}}(Q)$.

4 Example

As an example we describe a new algorithm to compute the Tate pairing in this section.

Algorithm 5. *Signed-Radix-2 Miller's Algorithm*

Input: $r = (r_t = 1, 0^{h_{t-1}}, r_{t-1}, \cdots, 0^{h_0}, r_0)_2$, $P = (x_P, y_P)$, $Q = (x, y)$
Output: $f_{rP}(Q)^{(q^k-1)/r} \in \mathbb{F}_{q^k}^*$

1: $T = P$, $f = 1$;
2: For $i = t - 1$ *downto* 0 *do*
3: *if* h_i *is even, then use the Algorithm 2 to compute* $f_{4^{h_i/2}T}$ *and* $4^{h_i/2}T$; *set* $T = 4^{h_i/2}T$ *then to compute* $f_{T+P}(Q)$ *if* $r_{i+1} = 1$ *or* $f_{T-P}(Q) = f_T \frac{l_T}{l_{-T,P}}(Q)$ *if* $r_{i+1} = -1$;
4: *if* h_i *is odd, then use the Algorithm 2 to compute* $f_{4^{(h_i-1)/2}T}$ *and* $4^{(h_i-1)/2}T$; *set* $T = 4^{(h_i-1)/2}T$ *then to compute* $f_{2T+P}(Q)$ *if* $r_{i+1} = 1$ *or* $f_{2T-P}(Q)$ *if* $r_{i+1} = -1$;

5: EndFor
6: Return $f^{(q^k-1)/r}$.
———————————————————————————————

For an integer r, consider the signed radix-2 representation of r. This representation is advantaged for the non-zero digits is one-third of the length of the representation on average. We write the radix-2 representation of r as $r = (r_t = 1, 0^{h_{t-1}}, r_{t-1}, \cdots, 0^{h_0}, r_0)_2$, where the r_i is 1 or -1. The above algorithm is a modified Miller's algorithm with the signed radix-2 representation.

5 Conclusion

In this paper, several strategies to compute the Tate pairing efficiently are given. The concept of Miller's path which let us consider the scalar multiplications and the computation of line functions in the same time is proposed. A useful fact is stated in Lemma 1 and simple formulas to compute f_{4T}, f_{2T-P} and f_{6T} etc. are presented. Similar idea as in [12] is used to compute f_{3T} to simplify the computation. These algorithms are also used to compute $f_{3T\pm P}$, $f_{4T\pm P}$ and f_{6T+P}. Furthermore, these methods can also be applied to other algorithms for the computation of Tate pairings. In practical applications, the computation of f_{2^kT} or f_{3^kT} are also needed. Certainly, iterative method can be used to reduce some cost, but it only reduce the cost of scalar multiplications. So, how to simplify the formula of Miller's functions f_{2^kT} and f_{3^kT} are still crucial open problems.

Acknowledgements

The authors would like to thank anonymous referees for their valuable comments and suggestions.

References

1. P.S.L.M. Barreto, H.Y. Kim, B. Lynn and M. Scott. Efficient algorithms for pairing-based cryptosystems, Advances in Cryptology–Crypto'2002, LNCS 2442, 354-368, Spriger-Verlag, 2002.
2. P.S.L.M. Barreto, B. Lynn, and M. Scott. On the selection of pairing-friendly groups, SAC 2003, LNCS 3006, 17-25, Spinger-Verlag, 2004.
3. I.F.Blake, V.Kumar Murty and Guangwu Xu. Refinements of Miller's algorithm for computing the Weil/Tate pairing, J.Algorithms 58(2), 134-149, 2006.
4. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing, SIAM Journal of Computing 32, 586-615, 2003.
5. D. Boneh, B. Lynn and H. Shacham. Short signatures from the Weil pairing. Advances in Cryptology–Asiacrypt 2001, LNCS 2248, 514–532, Springer-Verlag, 2001.
6. M. Ciet, M. Joye, K. Lauter, and P.L. Montgomery. Trading inversions for multiplications in elliptic curve cryptography, Design, Codes and Cryptography 39(2), 189-206, 2006.

7. H. Cohen, A. Miyaji, and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates, Advances in Cryptology–ASIACRYPT'98, LNCS 1514, 51–65, Springer-Verlag, 1998.
8. S. Duquesne and G. Frey. Background on pairings, In Handbook of elliptic and hyperelliptic curve cryptography, Discrete Math. Appl., 115-124, Chapman & Hall/CRC, Boca Raton, FL, 2006.
9. S. Duquesne and G. Frey. Implementation of pairings, In Handbook of elliptic and hyperelliptic curve cryptography, Discrete Math. Appl., 389-404, Chapman & Hall/CRC, Boca Raton, FL, 2006.
10. S. Duquesne and T. Lange. Pairing-based cryptography, In Handbook of elliptic and hyperelliptic curve cryptography, Discrete Math. Appl., 573-590, Chapman & Hall/CRC, Boca Raton, FL, 2006.
11. R. Dutta, R. Barua and P. Sarkar. Pairing-based cryptography: a survey, Cryptology ePrint Archive, Report 2004/064, 2004.
12. K. Eisenträger, K. Lauter and P.L. Montgomery. Fast elliptic curve arithmetic and improved Weil pairing evaluation, CT-RSA 2003, LNCS 2612, 343-354, Springer-Verlag, 2003.
13. S. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing, LNCS 2369, 324-337, Springer Verlag, 2002.
14. S. Galbraith. Pairings, Advances in elliptic curve cryptography, London Math. Soc. Lecture Note Ser. 317, 183–213, Cambridge Univ. Press, 2005.
15. Robert Granger, Dan Page and Nigel Smart. High security pairing-based cryptography revisited, LNCS 4076, 480–494, Springer Verlag, 2006.
16. D. Hankerson, A.J. Menezesand S. Vanstone, Guide to Elliptic Curve Cryptography, Springer Verlag, 2004.
17. T. Izu and T. Takagi. Efficient computations of the Tate pairing for the large MOV degrees. ICISC'02, LNCS 2587, 283-297, Springer-Verlag, 2003.
18. N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels, Cryptology ePrint Archive, Report 2005/076, 2005.
19. A.J. Menezes. Elliptic Curve Public Key Cryptosystems, Kluwer Academic Publishers, 1993.
20. V. Miller. Short programs for functions on curves, unpublished manuscript, 1986.
21. V. Miller. The Weil pairing, and its efficient calculation, J. Cryptology 17(4), 235-261, 2004.
22. R. Sakai, K. Ohgishiand M. Kasahara. Cryptosystems based on pairing, SCIS'00, no. C20, 2000.
23. M. Scott. Computing the Tate pairing, CT-RSA, Feb., 2005, San Francisco, LNCS 3376, 293-304, Springer-Verlag, 2005.
24. M. Scott, N. Costigan and W. Abdulwahab. Implementing cryptographic pairings on smartcards, Cryptography ePrint Archive, Report 2006/144, 2006.
25. J.H. Silverman. The Arithmetic of Elliptic Curves, GTM 106, Springer-Verlag, Berlin, 1986.